LARGE-DOMAIN DISCRETE INTEGRATION TECHNIQUES FOR THE WAVE EQUATION AS AN AID IN THE CALCULATION OF PROPAGATION LOSS AND THE STUDY OF ADAPTIVE ACOUSTIC ARRAYS

by

Roy Martin Johnson

United States
Naval Postgraduate School

THESIS

LARGE-DOMAIN DISCRETE INTEGRATION TECHNIQUES
FOR THE WAVE EQUATION AS AN AID
IN THE CALCULATION OF PROPAGATION LOSS
AND THE STUDY OF ADAPTIVE ACOUSTIC ARRAYS

by

Roy Martin Johnson

October 1969

Large-Domain Discrete Integration Techniques for the Wave Equation as an Aid in the Calculation of Propagation Loss and the Study of Adaptive Acoustic Arrays

by

Roy Martin Johnson
B.S., University of California, Berkeley, 1954

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

from the
NAVAL POSTGRADUATE SCHOOL
October 1969

ABSTRACT

This work is concerned with two inter-related problems: (1) the theory and application of algorithms for the discrete integration of the acoustic wave equation in large, multi-dimensional, variable-parameter domains and, (2) the automatic synthesis of adaptive acoustic arrays for the detection and estimation of received acoustic signals which are incident from such domains.

It is shown that the steady-state integration of large domains may be partitioned into subdomain integrations for slowly-varying variable-parameter domains. A new method of integration based upon the Fast Fourier Transform is given. A new method for obtaining closed form coefficient expressions for the Fast Fourier Transform is shown and illustrated.

The results of a general computer-display simulation program for the Widrow feed-forward algorithm are given. Several limitations and possible modifications to the Widrow procedure are given. The use of Kalman-type filters as an alternative method is introduced.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF SYMBOLS

Symbol	Description
$\varkappa(x)$	Bulk modulus of elasticity
ρ	fluid mass density
$c(x)$	speed of sound
$p(x,t)$	pressure
t	time
x,y,z	cartesian spacial coordinate
$\underline{d}(\underline{x},t)$	vector relative particle displacement
$\underline{v}(\underline{x},t)$	vector velocity
α	normalization constant; attenuation coustant
β	normalization constant; phase constant
$\underline{n}$	normal unit vector
$G(f)$	signal frequency spectrum
B	signal spectrum bandwidth
α	heat capacity
η	heat conductivity
T_m^n	temperature at position $m\Delta x$ at time $n\Delta t$
τ	discrete expansion time growth constant
Δt	discrete time increment
Δx	spacial grid increment
ζ_x	stability factor for explicit algorithm prediction

H_l	Fourier Transform prediction stability matrix
G	implicit integration scheme stability gain matrix
$\underline{u}_m^o$	implicit integration scheme state vector
$\hat{p}_k^o$	FFT pressure wavenumber coefficients
G_{10}	implicit scheme FFT wavenumber gain matrix
$\delta_{k,\ell}$	Kronecker delta function
$\underline{u}_h$	homogeneous solution state vector
$\underline{u}_i$	inhomogeneous solution state vector
γ	complex propagation constant
ω	angular frequency
f_c	discrete-model cutoff frequency
Z_o	characteristic wave impedance
$Z(x)$	complex wave impedance
R	series resistance / length of transmission line
L	series inductance / length of transmission line
G	shunt conductance / length of transmission line
C	shunt capacitance / length of transmission line
Z_s	series impedance / length of transmission line
Y_{sh}	shunt admittance / length of transmission line
$R(x)$	voltage reflection coefficient
$\hat{P}(x)$	pressure phasor
$\hat{V}(x)$	velocity phasor
T_{micro}	micro domain kinetic energy

V_{micro}	micro domain potential energy
U_{macro}	total macro domain internal energy
$G(k)$	discrete signal spectrum
$g(n)$	discrete time signal
$\delta_N(s-\ell)$	periodic Kronecker delta
$Us_N(s-\ell)$	periodic Heaviside function
$\underline{x}(t)$	signal input vector
$\underline{y}(t)$	signal output vector
$\underline{m}(t)$	observation transformation vector
$\underline{x}'(k)$	discrete input signal spectrum vector
$\sigma'(k)$	discrete desired input signal spectrum
$\underline{\mu}'(k)$	discrete observation transformation spectrum vector
$\underline{n}(t)$	input noise excitation vector
$\underline{\eta}'(k)$	input noise spectum vector
$\underline{R}(k)$	input noise unto-correlation vector
$\underline{P}(k)$	input power spectral density vector
$\zeta(T)$	test statistic for detection problem
$p(\underline{\psi}/\underline{\eta})$	conditional probability of input signal
$\underline{\theta}(k)$	"pre-whitening" spectral vector
$\Omega(k)$	spectral output response
λ	wavelength
a_i	feedback coefficients
b_i	feed-forward coefficients

z	Z-transform complex variable representing unit delay
$\underline{x}(z)$	Z-transform of array output vector
$G_i(z)$	Z-transform transfer function for i-th array element.
$\underline{w}(t)$	feed-forward weight vector
$\epsilon(k)$	discrete quadratic error signal
$H(k)$	observation matrix
$\Delta(k,k-1)$	excitation noise transition matrix
$\Phi(k,k-1)$	state transition matrix
$G(k)$	Kalman filter gain matrix
$\hat{s}(k/k)$	best conditional estimate of desired input signal given k previous estimates

ACKNOWLEDGEMENT

The author wishes to acknowledge the sincere and untiring efforts of Dr. Harold A. Titus in the guidance and direction of this research.

The author also wishes to echo with humbleness the Prayer of Thanksgiving by R. C. K. Lee:

> "Before God, all men are equally foolish, for we cannot begin to comprehend all the infinite mysteries surrounding us. If there be a grain of wisdom in this work, may it be to His glory, for He alone hath provided me with this splended opportunity for learning, the wonderful parents, teachers and friends, a harmonious home, excellent health -- and most of all, for the strength ever-sufficient from day to day."

I. INTRODUCTION

The detection, estimation, and distribution of acoustic energy signals propagating in inhomogeneous seawater are subjects of continuing interest. These subjects reach a focus of inter-relation in the design of efficient acoustic receiving arrays; such arrays should match the propagation solution properties of the location where the receiver is to be placed.

Past investigations into the trajectory and strength of acoustic signals in seawater have relied almost totally upon minimum-time plane-wave ray-path techniques or approximate normal mode solutions to crude stratification models. The theoretical and computational economy of these methods is their main asset. Rarely is a single model sufficient to represent faithfully the propagation characteristics of long acoustic paths involving multi-dimensional energy spreading or inhomogeneous medium phenomena such as convergence zones.

A more fundamental approach to such problems is the direct integration of the appropriate variable-coefficient wave partial-differential equation with any receiving array present being treated as part of boundary constraints. Unfortunately, no general theoretical methods exist for treating variable-coefficient partial-differential equations. However, foreseeable order of magnitude increases in digital computation speed and memory capacity make the direct finite-difference numerical integration of the wave equation a distinct possibility.

This paper attempts to make contributions to both the numerical integration techniques relative to solution of the large-domain variable-parameter acoustic wave equation and the theory and practice of adaptive acoustic antenna arrays.

In chapter II the acoustic wave partial-differential equation with variable coefficients is derived in properly posed form from first principles assuming infinitesimal deformations and that the coefficients are slowly varying and time independent. The appropriate boundary conditions are also discussed with particular emphasis being devoted to the matched boundary condition.

In chapter III methods for finite difference integration of discrete models for partial-differential equations are introduced. The problems of instability and convergence are illustrated for the one-dimensional diffusion equation which is the simplest partial-differential equation of common interest which has been extensively studied.

Continuous-discrete, explicit, and implicit solution algorithms are then studied and the results of actual computer calculations relative to required computation time, accuracy and stability are presented and discussed. Along with these results, a new method for semi-implicit integration of the three-dimensional wave equation using the Fast Fourier Transform algorithm of Cooley and Tukey is given. The method appears faster than all other methods tested with the exception of the successive approximation tri-diagonal method of Douglas and Gunn.

The magnitudes of the spacial increments Δx, Δy, Δz for proper wave equation simulations are obviously highly important; it is shown that the highest desired propagation frequency places a maximum value on these increments which is considerably less than the appropriate Shannon restriction.

The problem of integrating much larger spacial domains than could possibly be contained in any reasonable computer memory is taken up in chapter IV. General transient problem integrations appear to be very difficult to treat in such cases; the breaking up of the large integration domain into smaller memory requirement domains which can be integrated is frustrated in the variable parameter case by the question of which small domain is to be integrated first. It is shown however, that the steady state solution problem can be treated in such a manner for the slowly-varying parameter case by making use of the matched impedance boundary condition of ordinary transmission line theory. Since the problem of calculating steady state acoustic propagation loss is of this type the result is significant. The chapter concludes by presenting the results of several computer wave-equation integration problems.

The Fast Fourier Transform algorithm is a very efficient digital computer algorithm for computing the discrete Fourier transform of a finite-length sampled input function. Although a few discrete finite transforms can be obtained in closed form by use of the geometric sum formula, no general techniques for obtaining the explicit closed-form coefficients presently are known. In chapter V a new technique for obtaining such forms is given and two examples are worked as illustrations. The method is based on

taking first-order backwards finite differences and is quite general.

The subject of acoustic detection and estimation arrays is taken up in chapter VI. From the aspect that the local medium propagation properties may vary with the time and as a consequence the most efficient detector should also, the concept of automatic self-adapting arrays appear especially appealing. Therefore, after initially considering arrays from the classical optimum detection point of view, adaptive arrays are studied. A quite sophisticated general purpose interactive-display digital computer program has been used to study several possible automatic synthesis adaptive antenna arrays which may be placed on the boundary of a large wave-equation integration domain. General results are discussed and particular emphasis is devoted to possible modifications of existing adaption algorithms in order to increase their adaption speed.

II. THE PROPAGATION EQUATIONS

A. THE UNBOUNDED MEDIUM EQUATIONS

The problem of propagation of acoustic waves in salt water will be assumed to satisfy an acoustic wave equation; however, the solution for domains with variable parameters in more than one dimension is very complicated in general so that it is advisable to begin the analysis with a derivation of the medium equations from first principals in order to make clear the assumptions involved.

For infinitesimal acoustic disturbances in fluids the volume bulk modulus $\varkappa$ is related to the sound speed c by the relation[1]

$$\varkappa = \rho c^2 \tag{1}$$

where ρ is the density of the fluid. For the range of conditions dealt with here, the density ρ will be considered to be a constant. The acoustic speed c, however, is a function of both temperature and pressure.[2] Notwithstanding these complications, it is usual to assume that the resulting equation of state

$$c = c\ (p,t) \tag{2}$$

can be reduced to a parametric, single-valued function of position

$$\begin{aligned} c &= c\ (x,y,z) \\ &= c\ (\underline{x}) \end{aligned} \tag{3}$$

which is constant with respect to time for periods long compared with the periods of oscillations of interest.[2] By Hookes law,[1] the compression of particles relative to an increment change in pressure, p(x,y,z,t) is

$$p\ (x,y,z,t) = - \varkappa\ \nabla \cdot \underline{d}\ (x,y,z,t) \tag{4}$$

where $\underline{d}(\underline{x},t)$ is the vector relative displacement of the particles. Taking the partial derivative of (4) with respect to time gives

$$\frac{\partial p}{\partial t} = -\varkappa \nabla \cdot \underline{v}(\underline{x},t) \tag{5}$$

where $\underline{v}(\underline{x},t)$ is the vector acoustic disturbance velocity. The two dependent variables $p,\underline{v}$ are further related by Newton's second law which relates the vector acceleration of a sample mass volume to the pressure gradient

$$\rho \frac{d\underline{v}}{dt} = -\nabla p . \tag{6}$$

It is usual to further simplify (6) by assuming that in the total velocity derivative

$$\frac{d\underline{v}}{dt} = \frac{\partial \underline{v}}{\partial t} + (\nabla \cdot \underline{v})\underline{v} \tag{7}$$

the second term is of second order for all but the very highest possible frequencies so that

$$\rho \frac{\partial \underline{v}}{\partial t} = -\nabla p . \tag{8}$$

The two equations (5) and (8) constitute the fundamental first-order partial-differential equations to be solved in the absence of dissipative losses. The effects of dissipative losses usually are included empirically after solution of the lossless equations (5) and (8).[1]

Equation (8) is separable into three scalar partial-differential equations so that with (5) there are a total of four first-order partial differential equations to be solved:

$$\frac{\partial p}{\partial t} = - \kappa \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} \right) \tag{9}$$

$$\frac{\partial v_x}{\partial t} = - \frac{1}{\rho} \frac{\partial p}{\partial x} \tag{10}$$

$$\frac{\partial v_y}{\partial t} = - \frac{1}{\rho} \frac{\partial p}{\partial y} \tag{11}$$

$$\frac{\partial v_z}{\partial t} = - \frac{1}{\rho} \frac{\partial p}{\partial z} \quad . \tag{12}$$

The bulk modulus κ is a function of position since by (3) c is a function of position and ρ has been assumed constant. The set of equations (9,10, 11, 12) are a mixed set of equations in that they are functions of two dependent variables p and $\underline{v}$. They may be unmixed by taking the partial of (9) with respect to t,

$$\frac{\partial^2 p}{\partial t^2} = - \kappa \left(\frac{\partial}{\partial t} \left(\frac{\partial v_x}{\partial x} \right) + \frac{\partial}{\partial t} \left(\frac{\partial v_y}{\partial y} \right) + \frac{\partial}{\partial t} \left(\frac{\partial v_z}{\partial z} \right) \right) \tag{13}$$

which assuming continuity of first derivatives may be written as

$$\begin{aligned} \frac{\partial^2 p}{\partial t^2} &= - \kappa \nabla \cdot \left(\frac{1}{\rho} \frac{\partial \underline{v}}{\partial t} \right) \\ &= \frac{\kappa}{\rho} \nabla \cdot \nabla p \\ &= c^2 \nabla^2 p \quad . \end{aligned} \tag{14}$$

This is a scalar wave equation for the pressure. The velocity $\underline{v}$ satisfies an associated type of equation; if the partial of (8) with respect to t is taken and continuity is again assumed,

$$\begin{aligned} \frac{\partial^2 \underline{v}}{\partial t^2} &= - \frac{1}{\rho} \nabla \left(\frac{\partial p}{\partial t} \right) \\ &= - \frac{1}{\rho} \nabla \left(- \kappa \nabla \cdot \underline{v} \right) \\ &= \frac{\kappa}{\rho} \nabla (\nabla \cdot \underline{v}) \nabla \kappa \quad . \end{aligned} \tag{15}$$

The problems of interest here will be confined to unbounded fluids which are assumed unable to support shear stress waves; such being the case the curl of $\underline{v}$ must vanish and hence by the vector identity

$$\nabla(\nabla \cdot \underline{v}) = \nabla \times (\nabla \times \underline{v}) + \nabla^2 \underline{v} \tag{16}$$

(15) becomes

$$\frac{\partial^2 \underline{v}}{\partial t^2} = \frac{\varkappa}{\rho} \nabla^2 \underline{v} + \frac{1}{\rho} (\nabla \cdot \underline{v}) \nabla \varkappa \ . \tag{17}$$

The second term in (17) is usually neglected as being of second order since $\nabla \cdot \underline{v}$ is assumed small for the reason previously stated and in addition $\nabla \varkappa$ can be made also small by proper space partitioning in computer analyses. In ray-path solutions the second term can become important in certain cases.[5]

As previously stated, the equations (9,10,11,12) are fundamental; the derived set (14) and (17) are no less valid but their domain of solutions is larger.[4] It is possible for certain pathological solutions to exist satisfying (14) and (17) under a given set of boundary conditions but these solutions none-the-less will not satisfy (9,10,11,12). Thus only (9,10,11,12) are both necessary and sufficient. The pathological solutions to (14) and (17) will almost never occur in practice if (14) and (17) are solved analytically since the investigator will invariably pick characteristic forms in separation procedures which are physically regular in behavior; the problem is not academic however in computer solutions of the equations since poorly defined (real world) boundary conditions could possibly lead to successive approximate solutions converging to these pathological solutions. For all of these reasons the integration techniques in this work

will apply to the equations (9,10,11,12).

The set of equations (9,10,11,12) is also properly posed, i.e., a norm can be defined in an appropriate Banach space which is bounded.[4]

In implicit algorithm solutions to (9,10,11,12), it is very convenient to introduce new variables to make the equations (5) and (8) completely symmetric. Let

$$\underline{v} = \alpha \underline{v}' \tag{18}$$

$$p = \beta p' \tag{19}$$

then to first order

$$\frac{\partial p'}{\partial t} = - c^2 \nabla \cdot \underline{v}' \tag{20}$$

$$\frac{\partial \underline{v}'}{\partial t} = - \nabla p' \tag{21}$$

where

$$\beta = 1 \quad = \text{constant} \tag{22}$$

$$\alpha(\underline{x}) = \frac{1}{\rho c(\underline{x})} \quad . \tag{23}$$

The first order assumption in (20) and (21) assumes that $\alpha(\underline{x})$ is a slowly varying function of position. In addition to symmetry the new variables have a much reduced dynamic range which considerably increases the accuracy of computer computations.

B. BOUNDARY CONDITIONS

The equations (5) and (8) require a complete, consistent set of boundary conditions in order to guarantee that their solution will be unique and properly posed. The usual discrete methods of solution, from a practical if not a mathematical point of view, almost always are limited to bounded

domains. This excludes several important practical problems of interest including wave propagation in semi-infinite media. For the special case in which the steady-state propagation properties are desired it will be shown that by an appropriate spacial partitioning of the volume domain of interest together with an extension of the matched boundary condition of one-dimensional transmission line theory this problem may be solved.

The boundary condition at a fixed wall requires that the normal component of velocity vanish

$$\underline{n} \cdot \underline{v}\,(\text{bnd},t) = 0 \tag{24}$$

where $\underline{n}$ is a unit normal to the boundary surface.

An equivalent, not independent, condition on the pressure is from (6)

$$\underline{n} \cdot \nabla p \Big|_{\text{bnd}} = 0 \,. \tag{25}$$

At a free surface, if surface tension forces and atmospheric pressure may be ignored, the pressure must vanish

$$p\,(\text{surface},\, t) = 0 \,. \tag{26}$$

The matched boundary condition relates the pressure and velocity at a point and is an example of a mixed boundary condition. The condition is most fundamentally applied to one dimensional problems and is precisely reflectionless for constant parameter applications. The condition is

$$\left|\frac{p}{\underline{v}}\right| = \sqrt{\varkappa\rho} \tag{27}$$

where $|\underline{v}|$ represents the magnitude of the velocity. The condition can be approximately extended to variable-parameter media as well as multiple dimensions.

C. COORDINATE SYSTEMS

The equations (5) and (8) are in vectorial form even though derived from rectangular coordinate considerations; they hence hold true for any orthogonal curvilinear coordinate system. The choice of coordinate system depends upon the type of problems to be investigated as well as the domain size and approximations to be utilized. For example, physical problems inherently possess propagation properties which are three-dimensional, but frequency, source location, parameter variations and fixed boundaries may allow the propagation process to be confined to only one or two dimensions.

The discussions in this work are confined to rectangular and cylindrical coordinates but the technique often can be extended in a simple manner to spherical coordinates. The theory is usually illustrated for simplicity in one dimension but the computer simulations that have been carried out are always at least two-dimensional.

D. THEORETICAL SOLUTIONS TO THE PROPAGATION EQUATIONS

To measure the accuracy of computer solutions to (5-8), theoretical solutions are desirable. In those cases where the parameters are constants, the ordinary theoretical acoustic solutions to the wave equation in homogeneous media suffice; in the general case of inhomogeneous media exact theoretical solutions are almost non-existent except for some very special cases. The case of linear periodic-parameter media has received extensive treatment[5] and constitutes the most useful theoretical solution known as a comparison standard for finite discrete models of the wave

equation. In chapter III it is shown that such models are inherently dispersive, whereas the true wave equation in unbounded media is perfectly non-dispersive. However, if the partitioning is sufficiently small, the low frequency or Rayleigh limit approaches precisely the non-dispersive limit.

The step function is used as a test function in this work to verify that the Rayleigh limiting velocity is approached and that dispersive waveform distortion is not prohibitive.

Finite difference approximations to the wave equation are band-limited inherently so that the infinite slope at the discontinuity of a step function will never be represented exactly in computer solutions. The maximum attainable slope of the solution is most easily found by use of the Fourier transform.[6] Assume that the resulting solution at some arbitrary point $\underline{a}$ and time t is band-limited to $|f| < B$ hertz. Then by the Fourier transform the resulting pressure is represented by

$$p(\underline{a}, t) = \int_{-B}^{B} G(\underline{a}, f) e^{j2\pi ft} \, df \tag{28}$$

where $G(\underline{a}, f)$ is the spectral resolution function. Hence

$$\left|\frac{\partial p}{\partial t}\right| \leq 2\pi B \cdot \text{Max}\left(|G(\underline{a}, f)|\right) \cdot 2B \, . \tag{29}$$

Assuming that the pressure wave is an outward-going wave in the x direction only then the velocity in the x direction, by (27) is directly proportional to the pressure and hence by (9)

$$\frac{\partial p}{\partial t} = -\varkappa \frac{\partial v_x}{\partial x} \tag{30}$$

$$= -\varkappa \frac{1}{\sqrt{\varkappa \rho}} \frac{\partial p}{\partial x}$$

or

$$\left|\frac{\partial p}{\partial t}\right| = \sqrt{\frac{\rho}{\chi}}\left|\frac{\partial p}{\partial t}\right| \leq \sqrt{\frac{\rho}{\chi}} \cdot 4\pi B^2 \operatorname{Max}\left(\left|G(\underline{a},f)\right|\right) . \tag{31}$$

Thus the frequency band-limitedness of discrete models of the wave equation has as a direct consequence the result that the spacial representation of waveforms is also spacial-slope limited. In essence, then, transient-solution curves for step-function time input to discrete models of the wave equation will exhibit finite-maximum spacial-slope properties as the solution is observed at various fixed times. This has been observed in computer solutions and will be discussed in the chapter on discrete models for the wave equation.

III. DISCRETE INTEGRATION

A. THE DIFFUSION EQUATION

The purpose of this chapter is to introduce discrete integration models for partial differential equations suitable for programming on digital, analog, or hybrid computing machines. The relevant theory of such models is extensive. The heat diffusion equation in one space dimension is the simplest partial-differential equation possessing known theoretical solutions and which also illustrates the problems of instability and convergence common to discrete difference methods of solution. The wave equation, being a hyperbolic partial differential equation of the second order, is inherently more difficult to solve than the diffusion equation which is parabolic and of first order so that the relevant theory will be illustrated for the diffusion equation with necessary modifications for the wave equation being taken up in section C.

1. Theoretical Solution

The diffusion of heat through a one-dimensional solid of thermal conductivity η and heat capacity α is given by the parabolic partial differential equation

$$\alpha \frac{\partial T}{\partial t} = \frac{\partial}{\partial x} \left(\eta \frac{\partial T}{\partial x} \right) \tag{32}$$

where $T(x,t)$ is the temperature of the solid at position x and time t. Both the heat capacity and conductivity can in general be functions of position and time; it shall be assumed here that they are constants so that (32) becomes

$$\frac{\partial T}{\partial t} = \sigma \frac{\partial^2 T}{\partial x^2} \tag{33}$$

where

$$\sigma = \frac{\eta}{\alpha} \cdot \tag{34}$$

For a domain assume the real line $o \leq x \leq \pi$ and that $t \geq o$.

Further assume the given boundary values

$$T(x,o) = \varphi(x) \tag{35}$$

and

$$T(o,t) = T(\pi, t) = o. \tag{36}$$

As an example suppose

$$\begin{aligned} \varphi(x) &= cx \quad , \quad o \leq x \leq \frac{\pi}{2} \\ &= c(\pi - x), \ \frac{\pi}{2} \leq x \leq \pi \ . \end{aligned} \tag{37}$$

The theoretical solution to this boundary value problem is obtainable in convergent infinite Fourier series form; the assumed solution

$$T(x,t) = \sum_{m=-\infty}^{\infty} A_m(t) e^{jmx} \tag{38}$$

upon substitution in (33) requires that

$$\sum_{m=-\infty}^{\infty} \left\{ \frac{dA_m}{dt} - \sigma (jm)^2 A_m \right\} e^{jmx} = o \tag{39}$$

which can only be true for arbitrary values of x provided

$$\frac{dA_m}{dt} + m^2 \sigma A_m = o \ . \tag{40}$$

The solution to (40) which is bounded for all $t \geq o$ is

$$A_m(t) = A_m(o)\, e^{-\sigma m^2 t} \ . \tag{41}$$

Hence (38) may be written as

$$T(x,t) = \sum_{m=-\infty}^{\infty} A_m(o) e^{jmx - m^2 \sigma t} \tag{42}$$

The coefficients $A_m(o)$ are obtained by the usual coefficient formula from the distribution of temperature at $t = 0$:

$$A_m(o) = \int_{-\pi}^{\pi} \varphi(x)\, e^{-jmx}\, dx$$

$$= o\ , \qquad m \text{ even}$$

$$= \frac{2jC}{m^2\pi}\ , \qquad m \text{ odd}\ . \tag{43}$$

This theoretical solution has been evaluated for different values of the time by summing the series (42) with coefficients (43) on an IBM 360/67 to six decimal places; the solution is shown in Fig. 1.

The use of the Fourier series method of solution is limited strictly to linear constant-coefficient partial-differential equations whereas the finite difference methods to follow can be applied to much more general types of problems; the theoretical solution however does give a standard of comparison for any proposed finite-difference integration scheme when such a scheme is applied to linear constant-coefficient problems.

2. Discrete-Difference Equation - Instability

A great number of possible discrete models can be proposed to integrate numerically the equation (33) with the domain of (35,36). The majority of methods discretize x and t into some kind of grid space in order to convert the solution of the partial-differential equations or a set of algebraic equations. The grid may be uniform or non-uniform in the coordinates $m\Delta x$, $n\Delta t$; the crucial questions to be answered for any scheme, however, usually are as follows:

A. Is the integration scheme stable as the number of points in the $m\Delta x$, $n\Delta t$ solution space become infinite?

B. Does successive refinement of the solution by increasing the number of grid points in the solution space produce a solution which converges uniformly to the theoretical solution?

C. From a practical point of view the number of grid points must always remain finite. At what rate as $\Delta x, \Delta t \to o$ does the error between the finite difference approximation and the exact solution tend to zero?

The answers to these questions are exceedingly difficult in general; usually each case must be treated separately, especially in variable coefficient or non-linear problems. The known theory relative to questions A and B is fairly complete, especially for linear problems; the present status of knowledge about question C is quite inadequate since relative few exact solutions are known for variable coefficient problems.

The simplest and most commonly applied discrete-approximation scheme to the problem solved above is to discretize the space variable x into M sub-intervals Δx where

$$\Delta x = \frac{\pi}{M} \quad , \tag{44}$$

and then to successively iterate forward in time the M-1 equations

$$\frac{T_m^{n+1} - T_m^n}{\Delta t} = \frac{\sigma}{(\Delta t)^2} \left[T_{m+1}^n - 2T_m^n + T_{m-1}^n \right] , \tag{45}$$

$$m = 1, \ldots, M-1, \quad n = o, 1, 2, \ldots$$

The boundary conditions given in (36) become

$$T_o^n = T(o, n\,\Delta t) = o \qquad , n = o, 1, \ldots, \tag{46}$$

$$T_M^n = T(M\,\Delta x, n\,\Delta t) = o \qquad , n = o, 1, \ldots . \tag{47}$$

and the initial condition (37) prescribes

$$T_m^o = T(m\,\Delta x, o) = \varphi(m\,\Delta x) \qquad , \quad o \le m \le M . \tag{48}$$

Nothing has been said about the magnitude of the time increment Δt; this is the crux of the problem as is indicated in the plots of the computer solution to (45) shown in Figs. 1, 2(a,b,c). For small values of Δt the solution predicted by (43) agrees quite well with the theoretical solution shown in Fig. 1; when the value of Δt becomes large the computed solution becomes unstable as shown in Fig. 2(a,b,c). This is not caused by roundoff or truncation errors of the digital computer (double precision calculations have no effect) but is a direct result of the instabilities inherent in (45).

3. Exact Solution of the Difference Equations

Von Neumann[36] was the first investigator to successfully explain the instability inherent in the system (45) by means of Fourier analysis. To apply the method to (45) assume a series of the form

$$T_m^n = \sum_{k=-\infty}^{\infty} B_k \tau^n e^{jk(m\Delta x)} \tag{49}$$

is substituted in (45). After some algebra it can be seen that (49) satisfies the system provided

$$\tau(k) = 1 - \frac{2\sigma \Delta t}{(\Delta x)^2} (1 - \cos(k\Delta x)) . \tag{50}$$

Further, if the coefficients B_k are set equal to the corresponding $A_m(o)$ in (43), the solution (49) is exactly the same as the previous Fourier solution (42) except for the time dependence function. In (49) only the values at the actual mesh points $m\Delta x$, $n\Delta t$ of course have meaning. Assuming then, that both the Fourier solution (42) and the solution

$$T_m^n = \sum_{k=-\infty}^{\infty} A_k(o) \tau^n e^{jk(m\Delta x)} \tag{51}$$

are to be evaluated at the mesh points $m\Delta x$, $m = o,1,..,M$, the essential difference in the solutions occurs in the amplitude of the time factors $(\tau(k))^n$ and $e^{-\sigma m^2 t}$. The factors represent the time dependent performance of the individual Fourier harmonics in respective solutions and because of linearity are independent of each other. The exponential factor $e^{-\sigma m^2 t}$ clearly decays for increasing time whereas the factor

$$(\tau(k))^n = (1-k^2\sigma\Delta t + \frac{1}{12}k^4\sigma\Delta t(\Delta x)^2 - \ldots)^n \qquad (52)$$

tends to zero for increasing n only if

$$|\tau(k)| \leq 1 \quad . \qquad (53)$$

If for any harmonic the inequality (53) is not satisfied, that particular harmonic will grow as time increases leading to the instability shown in Fig. 2(a,b,c).

From (50) τ is real for all values of k and never more positive that + 1. The only possibility for instability thus must result from those cases where τ is less than -1. The worst condition occurs for the particular value of $k = k_c$ such that

$$\cos(k_c\Delta x) = -1, \qquad (54)$$

which for stability restricts

$$\frac{2\sigma\Delta t}{(\Delta x)^2} \leq 1 \quad . \qquad (55)$$

Conceptually, the equality in (54) occurs for those harmonics of wave-number k_c whose wave-lengths are of the order of magnitude of twice the discrete spacial grid spacing Δx. If these harmonics have amplitude exceeding the inequality restriction in (52) they will become unacceptably large as the time progresses. It has been shown that the inequality (55) is both necessary and sufficient for stability of this discrete integration scheme.

4. Refinement of Mesh - Convergence

If the domain of x does not change, it is clear that the magnitude of Δx will decrease as M increases, i.e., as the grid or mesh is refined. Question B previously stated in essence asks whether, as the grid is refined the limit

$$\lim_{M \to \infty} \left| T(m\Delta x, n\Delta t) - T_m^n \right| \to 0 \qquad (56)$$

holds true. In (56) $T(m\Delta x, n\Delta t)$ is the value of the temperature calculated by substituting the values $m\Delta x$, $n\Delta t$ in the exact theoretical solution of the diffusion equation. Hildebrand[37] has proven that the restriction (55) is sufficient to guarantee the limit (56). Further it can be stated that if a like analysis can be performed on other discretization algorithms for solution of the diffusion equation which lead to analogous expressions equivalent to (55) the sufficiency of convergence follows.

5. Rates of Convergence

The problem of rate of convergence (question C), usually must be handled on an individual basis and experience in actual computation is still the fundamental requirement.

In the case of the wave equation it will be shown in Chapter III that the minimum number of spacial increments required is set by the upper frequency limit for which the propagation properties are desired and by the characteristic acoustic velocity of the medium.

B. IMPLICIT ALGORITHMS AND THEIR SOLUTION

The restriction (55) on the relationship between Δt and Δx for stability is very severe; if Δx is refined by a factor of $\frac{1}{2}$, Δt must be

reduced to $\frac{1}{4}$ of its previous value to guarantee stability. The solution algorithm (45) is called an explicit algorithm in that it explicitly predicts T_m^{n+1} directly. Implicit solution algorithms are also available which require the solution of a set of simultaneous equations in order to find T_m^{n+1}. The more involved computations necessary are more than offset by the fact that the stability criterion may be much less severe, or in some cases, the system of equations may be unconditionally stable. In the latter cases the magnitude of the increment Δt is independent of Δx and can be set purely on the basis of the computing machine truncation error involved or the fineness of the final solution grid desired.

In general, if the partial differential equation is linear and only nearest-neighbor interactions are involved in the corresponding explicit algorithm, the choice of an implicit algorithm is indicated since the solution of the resulting linear equations involves inverting a nearly diagonal matrix.

Reference 4 lists a great number of implicit schemes for the diffusion equation, some of which are unconditionally stable. A typical example of an implicit scheme is the equation

$$\frac{T_m^{n+1} - T_m^n}{\Delta t} = \frac{\sigma}{(\Delta x)^2} \left[\beta \left(T_{m+1}^{n+1} - 2T_m^{n+1} + T_{m-1}^{n+1} \right) + (1-\beta) \left(T_{m+1}^{n} - 2T_m^{n} + T_{m-1}^{n} \right)\right] \quad (57)$$

where β is an adjustable parameter. If $\beta = 0$ the scheme reduces to the explicit scheme considered previously; if $\beta = \frac{1}{2}$ a centered time-difference equation results which is implicit. The equation (57) can be investigated as previously for stability; the condition equivalent to (55) is

$$\frac{2\sigma\Delta t}{(\Delta x)^2} \leq \frac{1}{1-2\beta} \quad , \quad \text{if } 0 \leq \beta < \tfrac{1}{2}$$
$$\text{no restriction} \quad , \quad \text{if } \tfrac{1}{2} \leq \beta \leq 1 \, . \tag{58}$$

Hence, for example if $\beta = \frac{1}{2}$ the equation is unconditionally stable.[7] It can be written in the form

$$-\zeta T_{m+1}^{n+1} + (2\zeta + \frac{1}{\Delta t}) T_m^{n+1} - \zeta T_{m-1}^{n+1} = (\frac{1}{\Delta t} - 2\zeta) T_m^n + \zeta T_{m+1}^n + \zeta T_{m-1}^n \tag{59}$$

where

$$\zeta = \frac{\sigma}{2(\Delta x)^2} \, . \tag{60}$$

The equation (59) contains M-1 unknowns, i.e., all of the variables on the left side of (59) minus the two known boundary conditions supplied by (46) and (47). The right side of (59) contains all known values so that the M-1 equations are sufficient for solution of the system by any of the standard methods applicable to the simultaneous solution of linear equations.

The equations (59) can be written in the general form

$$A \, \underline{T}^{n+1} = B \, \underline{T}^n \tag{61}$$

in which A and B are tri-diagonal matrices. For the one-dimensional case there exists a particularly efficient adaption of the Gauss elimination algorithm for the solution of this system which will be discussed in section C on the wave equation.

C. DISCRETE MODELS FOR THE WAVE EQUATION

The wave equation, in component symmetrical form given by (20) and (21), is inherently more difficult to solve than the diffusion equation considered in section A. To begin with it is of second order in the time

as well as in space and hence requires twice as many boundary values and at least twice as much computation to solve. Further it is a hyperbolic equation and the solutions need not be analytic for all values of the independent variables as is the case always for the diffusion equation.[39] The formulation (20) and (21) constitute a properly posed[4] system in that p' and $\underline{v}'$ constitute components of a general vector $\underline{u}$ which is dense and possesses finite norm in the appropriate Banach space.[8]

1. Continuous - Discrete Integration Approach

The simplest conceptual model for integrating the wave equation numerically is obtained by approximating the partial spacial derivatives occuring in (20) and (21) by finite differences. Although the approach is limited in digital applications by very severe restrictions on the time increment Δt in order that stability be maintained, the approach is fundamentally valid and easy to apply being successfully used by many investigators. To be efficient computationally the procedure requires a digital computer having a great deal of parallel processing capability or an analog type computer. A general hybrid machine of large capacity would probably be ideal but as yet the amount of operational amplifiers and electronic switches required have prohibited the economic construction of such a machine. With the advent in the near future of integrated circuit technology this approach may become dominant.

The result of replacing the partial spacial derivatives by finite difference approximations at a set of spacial grid points is to produce a set of ordinary-differential equations for the pressure and velocity at at these grid points. To illustrate the method assume in (20) and (21)

that the propagation is in the x direction with no variations in the y and z directions. The one-dimensional first-order equations in normalized form are then

$$\frac{\partial p}{\partial t} = - c \frac{\partial v_x}{\partial x} \tag{62}$$

$$\frac{\partial v_x}{\partial t} = - c \frac{\partial p}{\partial x} \tag{63}$$

where the superscripts have been dropped. The equations are completely analogous to the electrical transmission line equations [38] in which pressure is analogous to signal voltage and velocity is analogous to signal current. (In the un-normalized form of (62) the coefficient c would be replaced by the reciprocal of the series inductance/length.)

Conceptually the single variables p, v_x are replaced by column vectors $\underline{v}$ and $\underline{p}$

$$\underline{p}(t) = \begin{pmatrix} p_0(t) \\ \vdots \\ p_M(t) \end{pmatrix} \tag{64}$$

$$\underline{v}(t) = \begin{pmatrix} v_0(t) \\ \vdots \\ v_M(t) \end{pmatrix} \tag{65}$$

where

$$p_m(t) = p(m\,\Delta x, t) \tag{66}$$

$$v_m(t) = v_x(m\,\Delta x, t) \,. \tag{67}$$

If the domain of x is of length ℓ and it is assumed subdivided into M subintervals then $\underline{p}$ and $\underline{v}$ are M + 1 dimensional and the system of

equations (62), (63) are replaced by the 2(M - 1) equations,

$$\frac{dp_m}{dt} = - c(m\Delta x)\frac{v_{m+1} - v_{m-1}}{2\Delta x} \quad , \; m = 1, \ldots, M - 1 \quad (68)$$

$$\frac{dv_m}{dt} = - c(m\Delta x)\frac{p_{m+1} - p_{m-1}}{2\Delta x} \quad , \; m = 1, \ldots, M - 1 \quad (69)$$

and four equations for p_o, p_m, v_o, and v_m. These last variables are usually separated from (68) and (69) since their corresponding equation forms are dependent upon the appropriate boundary conditions.

The boundary conditions appropriate to the system (68), (69) are obviously 2(M + 1) in number: the 2(M + 1) initial values if the system is completely homogeneous or 2(M - 1) initial conditions for the equation (68), (69) and $p_o(t)$, $p_m(t)$, $v_o(t)$, $v_m(t)$ if the system is assumed excited by its boundary values. The solution to this system is ideally carried out on an analog computer using separate interconnected integrators for each of the variables $p_m(t)$, $v_m(t)$; the number of integrators is large however, and the setup and scaling problems involved are very cumbersome to handle for all but very small values of M. A hybrid machine with an automatic inter-connection matrix, potsetting and error control would be theoretically the best solution but no large machine meeting these requirements presently exists or can be dedicated to so restricted a set of problems. Thus digital computers are usually employed to sequentially integrate such systems. The process is slow (relative to true problem time) even for the fastest, large-core machines now in existence. However, it is accurate and refinement of grid procedure can be made automatic.

Digitally, any of the common methods [9] may be employed to integrate the ordinary differential equations (68), (69) but the accuracy is

clearly limited to the order of ($\Delta t + \Delta x$). The use of a small time increment Δt is also required since stability requires that $\Delta t/(\Delta x)^2$ be bounded as Δx, Δt tend to zero. The simplicity of the scheme together with the fact that only nearest-neighbor interactions are involved make the scheme quite useful if real-time computations are of secondary interest.

2. Explicit Scheme

An explicit scheme which is slightly more accurate as well as stable (provided ($c\,\Delta t/\Delta x) < 1$) is [10]

$$\frac{p_m^{n+1} - p_m^n}{\Delta t} = \frac{c}{\Delta x} \left(v_{m+\frac{1}{2}}^n - v_{m-\frac{1}{2}}^n\right) \tag{70}$$

$$\frac{v_{m-\frac{1}{2}}^{n+1} - v_{m-\frac{1}{2}}^n}{\Delta t} = \frac{c}{\Delta x} \left(p_m^{n+1} - p_{m-1}^{n+1}\right) . \tag{71}$$

Here fractional subscripts (purely a notational device) have been introduced in order to avoid double space intervals $2\,\Delta x$. The scheme (70, 71) can be arranged into purely explicit form as

$$p_m^{n+1} = p_m^n + \zeta \left(v_{m+\frac{1}{2}}^n - v_{m-\frac{1}{2}}^n\right) \tag{72}$$

$$v_{m-\frac{1}{2}}^{n+1} = v_{m-\frac{1}{2}}^n + \zeta \left(p_m^n - p_{m-1}^n\right) + \zeta^2 \left(v_{m-\frac{1}{2}}^n - 2v_{m-1}^n + v_{m-3/2}^n\right) \tag{73}$$

where

$$\zeta = \frac{c\,\Delta t}{\Delta x} . \tag{74}$$

This scheme has been extensively used and can be generalized to three dimensions. The result is the system

$$p(\ell, m,n,r+1) = p(\ell, m,n,r)$$
$$+ \zeta_x (v_x(\ell+\tfrac{1}{2}, m,n,r) - v_x(\ell-\tfrac{1}{2}, m,n,r))$$
$$+ \zeta_y (v_y(\ell+\tfrac{1}{2}, n,r) - v_y(\ell, m-\tfrac{1}{2}, n,r))$$
$$+ \zeta_z (v_z(\ell, m, n+\tfrac{1}{2}, r) - v_z(\ell, m, n-\tfrac{1}{2}, r)) \quad (75)$$

$$v_x(\ell-\tfrac{1}{2}, m,n,r+1) = v_x(\ell-\tfrac{1}{2}, m,n,r)$$
$$+ \zeta_x (p(\ell, m,n,r) - p(\ell-1, m,n,r))$$
$$+ \zeta_x^2 (v_x(\ell-\tfrac{1}{2}, m,n,r) - 2v_x(\ell-1, m,n,r)$$
$$+ v_x(\ell-3/2, m,n,r) \quad (76)$$

$$v_z(\ell, m-\tfrac{1}{2}, n,r+1) = v_y(\ell, m-\tfrac{1}{2}, n,r)$$
$$+ \zeta_y (p(\ell, m,n,r) - p(\ell, m-1,n,r))$$
$$+ \zeta_y^2 (v_y(\ell, m-\tfrac{1}{2}, n,r) - 2v_y(\ell, m-1,n,r)$$
$$+ v_z(\ell, m-3/2, n,r)) \quad (77)$$

$$v_z(\ell, m, n-\tfrac{1}{2}, r+1) = v_z(\ell, m, n-\tfrac{1}{2}, r)$$
$$+ \zeta_z (p(\ell, m,n,r) - p(\ell, m,n-1,r))$$
$$+ \zeta_z^2 (v_z(\ell, m,n-\tfrac{1}{2}, r) - 2v_z(\ell, m,n,-1,r)$$
$$+ v_z(\ell, m,n-3/2, r)) \quad (78)$$

where

$$p(\ell, m,n,r) = p(\ell\Delta x, m\Delta y, n\Delta z, r\Delta t),$$
$$\zeta_x = \frac{c\Delta t}{\Delta x}, \quad \zeta_y = \frac{c\Delta t}{\Delta y}, \quad \zeta_z = \frac{c\Delta t}{\Delta z}. \quad (79)$$

Computational experience on a CDC 6500 computer with this scheme has shown that: (1) The error is only slightly less than the scheme (67), (68); (2) The scheme is considerably more stable than (67), (68)

especially when the coefficients Δx, Δy, Δz are not equal and/or when the coefficient $c(\underline{x})$ varies appreciably. This is because a slight amount of predictor-corrector behavior is present in that an improved updated value of the pressure is used on the right hand sides of (76,77,78).

(3) The computation time is much longer than the generalization to three dimensions of (68,69).

In view of the experience (3) an implicit scheme is very desirable.

3. Solution of the Diffusion Equation Implicit System

In Section B an implicit scheme for solution of the diffusion equation was presented; the equations can be solved very efficiently as follows: Writing (57) in the form

$$-A_m T_{m+1}^{n+1} + B_m T_m^{n+1} - C_m T_{m-1}^{n+1} = D_m, \quad m = 1,\ldots,M-1 \tag{80}$$

it follows that

$$A_m = \frac{\sigma}{2(\Delta x)^2}, \quad B_m = \frac{\sigma}{(\Delta x)^2} + \frac{1}{\Delta t}, \quad C_m = A_m,$$
$$D_m = \left(\frac{1}{\Delta t} - \frac{\sigma}{(\Delta x)^2}\right) T_m^n + \frac{\sigma}{2(\Delta x)^2} T_{m+1}^n + \frac{\sigma}{2(\Delta x)^2} T_{m-1}^n . \tag{81}$$

In the above relations D_m is assumed completely known at all positions m at time state $n = o$. In addition, the conditions at $x = o$ and $x = M\Delta x$ are either assumed known at all times or they are assumed known at time $t = o$ and subsequently they satisfy a linear-discrete relationship equivalent to the usual Sturm-Louville problem. For the present, assuming the simplest case with homogeneous conditions (46) and (47), the system of equations (80) together with the left hand boundary condition (46) determine a one-parameter family of solutions, i.e., two sets of quantities

E_m and F_m are desired which for any member T_m^n satisfy the relation

$$T_m^{n+1} = E_m T_{m+1}^{n+1} + F_m . \tag{82}$$

If (82) is true for any member it is true for m = o which requires by virtue of (47) that

$$E_o = o, \quad F_o = o . \tag{83}$$

If now

$$T_{m-1}^{n+1} = E_{m-1} T_m^{n+i} + F_{m-1} \tag{84}$$

is substituted into (80) the result may be written as

$$T_m^{n+1} = \frac{A_m}{B_m - C_m E_{m-1}} T_{m+1}^n + \frac{D_m + C_m F_{m-1}}{B_m - C_m E_{m-1}} . \tag{85}$$

This in turn may be equated to (81) to obtain

$$E_m = \frac{A_m}{B_m - C_m E_{m-1}} , \quad m = 1,2,... \tag{86}$$

$$F_m = \frac{A_m}{B_m - C_m E_{m-1}} , \quad m = 1,2,... . \tag{87}$$

Starting with m = 1 and the values (83), the E_m and F_m can be calculated sequentially up to m = M-1. Once all the E_m, F_m are known (82) allows the calculation of T_m^{n+1} sequentially in decreasing m, (m = M-1, M-2, ..., 1) since for m = M the temperature T_{m+1}^{n+1} is known by (48).

This particular algorithm also has the advantage that the parameters A_m through F_m lie within a narrow dynamic range; it can be shown using the relations (81) that if $E_m \leq 1$ then

$$o \leq E_m \leq \frac{A_m}{B_m - C_m} < \frac{A_m}{A_m} = 1 .$$

Since by (83) $E_o = o$, E_m lies between o and 1. Further, since the solution of the parabolic diffusion equation is analytic the solution T_m^n must be bounded and this with the bound on E_m requires that F_m also be reasonably bounded by (82).

The greatest advantage of this particular implicit algorithm solution is the saving in computation; it requires only three multiplications and two divisions per space step, per time step. It is a direct result of the matrix A in (61) being tri-diagonal in form. The corresponding solution by direct matrix inversion, which would only need be done once if the coefficients are independent of n, would require in addition to the inversion procedure - the multiplication of a matrix by a column vector - which would be M multiplications per space step, per time step.

4. Solution of the One-Dimensional Wave Equation Implicit System

The above algorithm for the diffusion equation can be extended to the wave equation in certain cases. For the one-dimensional wave equation (62), (63) a symmetric implicit scheme is

$$\frac{p_m^{n+1} - p_m^n}{\Delta t} = \frac{c}{2\Delta x}\left(v_{m+\frac{1}{2}}^n - v_{m-\frac{1}{2}}^n + v_{m+\frac{1}{2}}^{n+1} - v_{m-\frac{1}{2}}^{n+1}\right) \quad (88)$$

$$\frac{v_{m-\frac{1}{2}}^{n+1} - v_{m-\frac{1}{2}}^n}{\Delta t} = \frac{c}{2\Delta x}\left(p_m^{n+1} - p_{m-1}^{n+1} + p_{m-1}^n - p_{m-1}^n\right) . \quad (89)$$

To investigate the stability of such schemes the Von Neumann theory has been generalized by Kreiss[11] and others.[11-15] Reference 4 summarizes the technique. Assuming that a suitable norm has been chosen for the domain of solutions of (88,,89) and that hence the Fourier analysis may

be applied to the system, let

$$\begin{bmatrix} p_m^n \\ v_m^n \end{bmatrix} = \sum_{k=-\infty}^{\infty} H_o(k) \begin{bmatrix} \hat{p}_k^n \\ \hat{v}_k \end{bmatrix} e^{jk(m\Delta x)} \tag{90}$$

$$\begin{bmatrix} p_m^{n+1} \\ v_m^{n+1} \end{bmatrix} = \sum_{k=-\infty}^{\infty} H_1(k) \begin{bmatrix} \hat{p}_k^{n+1} \\ \hat{v}_k^{n+1} \end{bmatrix} e^{jk(m\Delta x)} \quad . \tag{91}$$

If (90) is substituted into (88, 89) the result may be written as (after cancellation of the common factor exp (j k(m Δ x)) from all terms of the equations)

$$H_1 \begin{bmatrix} \hat{p}_k^{n+1} \\ \hat{v}_k^{n+1} \end{bmatrix} - H_o \begin{bmatrix} \hat{p}_k^{n} \\ \hat{v}_k^{n} \end{bmatrix} = o \tag{92}$$

where H_1 and H_o are the square matrices

$$H_1(k) = \frac{1}{\Delta t} \begin{bmatrix} 1 & , & -j\frac{c\Delta t}{\Delta x} \sin k\frac{\Delta x}{2} \\ -j\frac{c\Delta t}{\Delta x} c^{-jk\frac{\Delta x}{2}} \sin k\frac{\Delta x}{2} & , & e^{-jk\frac{\Delta x}{2}} \end{bmatrix} \tag{93}$$

$$H_o(k) = \frac{1}{\Delta t} \begin{bmatrix} 1 & , & j\frac{c\Delta t}{\Delta x} \sin k\frac{\Delta x}{2} \\ j\frac{c\Delta t}{\Delta x} e^{-jk\frac{\Delta x}{2}} \sin k\frac{\Delta x}{2} & , & e^{-jk\frac{\Delta x}{2}} \end{bmatrix} . \tag{94}$$

The gain matrix $G(k, \Delta x, \Delta t)$ is then defined as

$$G(k, \Delta x, \Delta t) = (H_1)^{-1} H_o. \tag{95}$$

and the general Von Neumann condition due to Richtmeyer[4] may be stated as as follows:

A sufficient condition for stability (and by the Lax equivalence theorem, for convergence) of a two-level difference scheme with transform matrices H_1 and H_o is that the gain matrix

$$\left[G(k, \Delta x, \Delta t)\right]^n \quad \text{for} \quad \left\{\begin{array}{l} o < \Delta t < \tau \\ o \leq n\Delta t \leq T \\ \text{all } k \text{ in } (90,91) \end{array}\right\} \tag{96}$$

be uniformly bounded.

The matrix G for the present problem can be written as

$$G(k, \Delta x, \Delta t) = \frac{1}{1+a^2/4} \begin{pmatrix} 1-a^2/4 & ja \\ ja & 1-a^2/4 \end{pmatrix} \tag{97}$$

where

$$a = \frac{2c\,\Delta t}{\Delta x} \sin k \frac{\Delta x}{2} \tag{98}$$

It satisfies

$$G^*G - GG^* = o \tag{99}$$

and is therefore unitary.[13] Hence G is always bounded and the scheme (88,89) is unconditionally stable. The system may be solved by the same scheme as previously shown for the diffusion equation since it is also tri-diagonal. In order to guarantee the non-singularity of the matrices involved it is easier to deal with integer subscripts; (88,89) can be written as

$$p_m^{n+1} - p_m^n = -\frac{c\,\Delta t}{4\,\Delta x} \left(v_{m+1}^n - v_{m-1}^n + v_{m+1}^{n+1} - v_{m-1}^{n+1}\right) \tag{100}$$

$$v_m^{n+1} - v_m^n = -\frac{c\Delta t}{4\Delta x}\left(p_{m+1}^n - p_{m-1}^n + p_{m+1}^{n+1} - p_{m-1}^{n+1}\right) . \quad (101)$$

The scalar equation (80) becomes the vector equation

$$-A_m \underline{u}_{m+1}^{n+1} + B_m \underline{u}_m^{n+1} - C_m \underline{u}_{m-1}^{n+1} = \underline{d}m \quad (102)$$

where

$$\underline{u}_m^{n+1} = \begin{pmatrix} p_m^{n+1} \\ v_m^{n+1} \end{pmatrix} , \quad \underline{d}m = \begin{pmatrix} p_m^n - \frac{c\Delta t}{4\Delta x}\left(v_{m+1}^n - v_{m-1}^n\right) \\ v_m^n - \frac{c\Delta t}{4\Delta x}\left(p_{m+1}^n - p_{m-1}^n\right) \end{pmatrix} \quad (103)$$

and

$$A = \begin{pmatrix} o & \frac{c\Delta t}{4\Delta x} \\ \frac{c\Delta t}{4\Delta x} & o \end{pmatrix} , \quad B = \begin{pmatrix} 1 & o \\ o & 1 \end{pmatrix} , \quad C = -A . \quad (104)$$

Again letting

$$\underline{u}_{m+1} = E_m \underline{u}_m + \underline{f}_m , \quad m = 0, 1, \ldots, M \quad (105)$$

the results

$$E_m = \left(B_m - C_m E_{m-1}\right)^{-1} A_m , \quad m = 1, \ldots, M-1 \quad (106)$$

$$\underline{f}_m = \left(B_m - C_m E_{m-1}\right)^{-1} \left(\underline{d}_m - C_m \underline{f}_{m-1}\right) , \quad m = 1, \ldots, M-1 \quad (107)$$

are obtained. To solve the system suppose the matched boundary condition is applied at $x = M\Delta x$ in the form

$$\begin{pmatrix} p_{m-1} \\ v_{m-1} \end{pmatrix} = \begin{pmatrix} o & -1 \\ 1 & o \end{pmatrix} \begin{pmatrix} p_m \\ v_m \end{pmatrix} . \quad (108)$$

and a homogeneous velocity condition at $x = o$ in the form

$$\begin{pmatrix} p_o \\ v_o \end{pmatrix} = \begin{pmatrix} p_o(n) \\ o \end{pmatrix} , \quad p_o(n) \text{ given.} \quad (109)$$

The "matched" condition (108) is clearly only an approximation to the correct condition for large Δx, but becomes progressively better as $\Delta x \to 0$.

This closed-trough problem has been programmed on the IBM 360 with the following results:

A. The computation time for equivalent Δx is reduced by an order of magnitude over the scheme (73,74)

B. The accuracy for a given Δx is considerably poorer than obtained by either (68,69) or (73,74) but this is thought to be because the boundary condition (108) is properly simulated only in the limit of $\Delta x \to 0$. Because of (A) above a much more rapid refinement of Δx is possible leading to very satisfactory results.

C. Variable parameters are routinely handled and no instabilities were observed for reasonable parameter variations.

5. Three-Dimensional Implicit Formulation; Fractional Step Algorithm.

The results obtained for the one-dimensional implicit scheme for the wave equation, particularly the increase in computing speed, make a three-dimensional implicit scheme highly desirable. Therefore, a straightforward generalization of (100,101) was considered. The resulting scheme is

$$
\begin{aligned}
&p(\ell,m,n,r+1) - p(\ell,m,n,r) \\
&= -\frac{\Delta t c(\ell,m,n,)}{4}\left[\frac{1}{\Delta x}\left(v_x(\ell+1,m,n,r) - (\ell-1,m,n,r)\right.\right. \\
&\qquad + v_x(\ell+1,m,n,r+1) - v_x(\ell-1,m,n,r+1) \\
&\quad + \frac{1}{\Delta y}\left(v_y(\ell,m+1,n,r) - v_y(\ell,m-1,n,r) + v_y(\ell,m+1,n,r+1)\right. \\
&\qquad \left. - v_y(\ell,m-1,n,r+1)\right) \\
&\quad + \frac{1}{\Delta z}\left(v_z(\ell,m,n+1,r) - v_z(\ell,m,n-1,r) + v_z(\ell,m,n+1,r+1)\right. \\
&\qquad \left.\left. - v_z(\ell,m,n-1,r+1)\right)\right] \qquad (110)
\end{aligned}
$$

$$v_x(\ell,m,n,r+1) - v_x(\ell,m,n,r) =$$

$$-\frac{\Delta t c(\ell,m,n)}{4\,\Delta x}\,(p(\ell+1,m,n,r) - p(\ell-1,m,n,r) + p(\ell+1,m,n,r+1) - p(\ell-1,m,n,r+1)) \tag{111}$$

$$v_y(\ell,m,n,r+1) - v_y(\ell,m,n,r) =$$

$$-\frac{\Delta t c(\ell,m,n)}{4\,\Delta y}\,(p(\ell,m+1,n,r) - p(\ell,m-1,n,r) + p(\ell,m+1,n,r+1) - p(\ell,m-1,n,r+1) \tag{112}$$

$$v_z(\ell,m,n,r+1) - v_z(\ell,m,n,r) =$$

$$-\frac{\Delta t c(\ell,m,n)}{4\,\Delta z}\,(p(\ell,m,n+1,r) - p(\ell,m,n-1,r) + p(\ell,m,n+1,r+1) - p(\ell,m,n-1,r+1)) \tag{113}$$

where as before $p(\ell,m,n,r) = p(\ell\Delta x, m\Delta y, n\Delta z, r\Delta t)$, etc. This algorithm is again unconditionally stable as $G(\underline{k}, \Delta x, \Delta y, \Delta z, \Delta t)$ is unitary but the tri-diagonal behavior has been lost; the matrix H_1 is of the form

$$H_1 = \begin{bmatrix} 1, & j\frac{c\Delta t}{2\Delta x}\sin k_1\Delta x, & j\frac{c\Delta t}{2\Delta y}\sin k_2\Delta y, & j\frac{c\Delta t}{2\Delta z}\sin k_3\Delta z \\ j\frac{c\Delta t}{2\Delta x}\sin k_1\Delta x, & 1, & 0, & 0 \\ j\frac{c\Delta t}{2\Delta y}\sin k_2\Delta y, & 0, & 1, & 0 \\ j\frac{c\Delta t}{2\Delta z}\sin k_3\Delta z, & 0, & 0, & 1 \end{bmatrix} \frac{1}{\Delta t} \tag{114}$$

which is penta-diagonal. Several problems were solved using this algorithm; it is highly accurate, stable under quite general conditions, but is relatively slow and requires a great deal of programming effort to use in the application being studied.

Many investigators have proposed schemes for solving the three-dimensional wave equation efficiently while maintaining stability. The

Soviet mathematicians Bagunovskii and Goduno,[16] and later[17] Yanenko have used partitions of the spacial operator for the various derivatives, applying each operator for fraction of the total time step Δt. Douglas and Gunn[18] proposed a very general scheme for any number of space dimensions which involves p-tridiagonal calculations for a p-th order cartesion system. Douglas and Gunn's paper contains examples for the diffusion equation; the procedure has been applied to the system (9,10,11, 12) of Chapter II under the symmetric condition $\Delta x = \Delta y = \Delta z$. The resulting system of equations -- which is quite long -- is shown in Appendix A. The system can be shown to be unconditionally stable if the coefficient c is a constant; for variable c considerable computational experice seems to indicate considerable margin of stability exists if the variations in c do not exceed five percent.

6. The Fast Fourier Transform Technique for Constant Coefficients

In the special case that the velocity c may be treated as a constant over the domain of integration, the use of the digital discrete Fast Fourier Transform (FFT) algorithm[19] provides a simple, elegant solution of either explicit or implicit algorithm schemes. The application of the discrete Fourier transform to this problem in the manner to follow is new.

Since implicit algorithms in general, as shown above, exhibit superior stability properties and when properly formulated greatly reduced computation requirements, the FFT algorithm was applied to their solution.

The implicit scheme (100,101) will be solved to illustrate the method. Assume that the initial states

$$\underline{u}_m^o = \begin{pmatrix} p_m^o \\ v_m^o \end{pmatrix} \quad , \quad m = o,1,\ldots,M-1 \tag{115}$$

are known and that the number of sample points M, including boundary points is an integral multiple of two. This restriction is not absolutely necessary but its imposition guarantees the maximum computational efficiency for the FFT algorithm. The discrete set of values may then be represented by the finite inverse Fourier transform as

$$\begin{pmatrix} p_m^o \\ p_m^o \end{pmatrix} = \sum_{k=o}^{M-1} \begin{pmatrix} \hat{p}_k^o \\ \hat{v}_k^o \end{pmatrix} e^{j \frac{2\pi k}{M} m} \tag{116}$$

where the spectral functions $\hat{p}_k^o$, $\hat{v}_k^o$ are obtained by the process

$$\begin{pmatrix} \hat{p}_k^o \\ v_k^o \end{pmatrix} = \frac{1}{M} \sum_{m=o}^{M-1} \begin{pmatrix} p_m^o \\ v_m^o \end{pmatrix} e^{-j \frac{2\pi m}{M} k} \quad . \tag{117}$$

This result, coupled with the fact that, by (116)

$$\begin{pmatrix} p_{m\pm\beta}^o \\ v_{m\pm\beta}^o \end{pmatrix} = \sum_{k=o}^{M-1} \begin{pmatrix} \hat{p}_k^o \\ \hat{v}_k^o \end{pmatrix} e^{j \frac{2\pi k}{M} (m\pm\beta)} \quad , \beta = \text{integer}, \tag{118}$$

leads to the following equation if the transform is applied to (100,101) under the assumption that c is a constant.

$$\frac{1}{\Delta t} \sum_{k=o}^{M-1} \left[\begin{pmatrix} 1 & -j \frac{c\Delta t}{2\Delta x} \sin \frac{2\pi k}{M} \\ -j \frac{c\Delta t}{2\Delta x} \sin \frac{2\pi k}{M} & 1 \end{pmatrix} \begin{pmatrix} \hat{p}_k^1 \\ \hat{v}_k^1 \end{pmatrix} - \begin{pmatrix} 1 & j \frac{c\Delta t}{2\Delta x} \sin \frac{2\pi k}{M} \\ j \frac{c\Delta t}{2\Delta x} \sin \frac{2\pi k}{M} & 1 \end{pmatrix} \begin{pmatrix} \hat{p}_k^o \\ \hat{v}_k^o \end{pmatrix} \right] e^{j \frac{2\pi k}{M} m} = o \tag{119}$$

This can be true for arbitrary m only if the large bracket term vanishes.

Hence defining the matrices

$$G_1(k, \Delta x, \Delta t) = \begin{pmatrix} 1 & -\alpha \\ -\alpha & 1 \end{pmatrix}, \tag{120}$$

and

$$G_o(k, \Delta x, \Delta t) = \begin{pmatrix} 1 & \alpha \\ \alpha & 1 \end{pmatrix}, \tag{121}$$

where

$$\alpha = j\frac{c\Delta t}{2\Delta x} \sin\frac{2\pi k}{M}, \tag{122}$$

it follows that

$$\begin{pmatrix} \hat{p}_k^1 \\ \hat{v}_k^1 \end{pmatrix} = (G_1)^{-1} G_o \begin{pmatrix} \hat{p}_k^o \\ \hat{v}_k^o \end{pmatrix}$$

$$= G_{10} \begin{pmatrix} \hat{p}_k^o \\ \hat{v}_k^o \end{pmatrix} \tag{123}$$

or

$$\begin{pmatrix} p_m^1 \\ v_m^1 \end{pmatrix} = \sum_{k=o}^{M-1} G_{10}(k, \Delta x, \Delta t) \begin{pmatrix} \hat{p}_k^o \\ \hat{v}_k^o \end{pmatrix} e^{j\frac{2\pi k}{M}m}. \tag{124}$$

Also, by induction

$$\begin{pmatrix} p_m^n \\ v_m^n \end{pmatrix} = \sum_{k=o}^{M-1} (G_{10}(k, \Delta x, \Delta t))^n \begin{pmatrix} \hat{p}_k^o \\ \hat{v}_k^o \end{pmatrix} e^{j\frac{2\pi k}{M}}. \tag{125}$$

Hence apart from the calculation of the powers of $G_{10}(k)$ the entire process involves only the application of the FFT algorithm at the time states $n=o$ and the desired state n. The powers of $G_{10}(k)$ may be most efficiently evaluated for large n by use of Sylvester's theorem.[20]

The procedure is readily extendable to three dimensions; the penta-diagonal scheme (110,111,112,113) has the solution

$$\underline{u}(\ell,m,n,r) = \begin{pmatrix} p(\ell,m,n,r) \\ v_x(\ell,m,n,r) \\ v_y(\ell,m,n,r) \\ v_z(\ell,m,n,r) \end{pmatrix}$$

$$= \sum_{k_3=o}^{N-1} \sum_{k_2=o}^{M-1} \sum_{k_1=o}^{L-1} (G_{10}(k_1,k_2,k_3))^r \begin{bmatrix} \hat{p}(k_1,k_2,k_3,o) \\ \hat{v}_x(k_1,k_2,k_3,o) \\ \hat{v}_y(k_1,k_2,k_3,o) \\ \hat{v}_z(k_1,k_2,k_3,o) \end{bmatrix}$$

$$\cdot e^{j2\pi \left(\frac{k_1 \ell}{L} + \frac{k_2 m}{M} + \frac{k_3 n}{N} \right)} , \qquad (126)$$

where

$$G_{10}(k_1,k_2,k_3) = G_1(k_1,k_2,k_3)^{-1} \; G_o(k_1,k_2,k_3)$$

$$= \begin{pmatrix} 1 & \alpha & \beta & \gamma \\ \alpha & 1 & o & o \\ \beta & o & 1 & o \\ \gamma & o & o & 1 \end{pmatrix}^{-1} \begin{pmatrix} 1 & -\alpha & -\beta & -\gamma \\ -\alpha & 1 & o & o \\ -\rho & o & 1 & o \\ -\gamma & o & o & 1 \end{pmatrix} , \qquad (127)$$

and

$$\alpha = \frac{jc\Delta t}{2\Delta x} \sin \frac{2\pi k_1}{L}, \quad \beta = \frac{jc\Delta t}{2\Delta y} \sin \frac{2\pi k_2}{M}, \quad \gamma = \frac{jc\Delta t}{2\Delta z} \sin \frac{2\pi k_3}{N} . \tag{128}$$

There is, of course, an unfortunate oversight in the above manipulations: the equations (100,101) involve the boundary conditions at m=o and m=M which are outside of the indexing of (100,101). The same must be true in the above formulation so that what has been solved is the system in which $p(\ell,m,n,o,), v(\ell,m,n,o)$ are known and the tacit homogeneous boundary conditions $p(-1,m,n,r) = p(L,m,n,r) = p(\ell,-1,n,r) = p(1,M,n,r) = o$, etc., have been assumed. (A homogeneous nearest-neighbor shell of boundary conditions on p,v enclosing the parallelepiped $L\Delta x$ by $M\Delta y$ by $N\Delta z$.) The method of solution can be extended to handle this case by making use of the theorem by Courant and Hilbert,[31] wherein it is shown that: homogeneous differential equations with non-homogeneous boundary conditions are essentially equivalent to non-homogeneous differential equations with homogeneous boundary conditions. Usually the application of the theorem requires in addition that certain smoothness conditions be satisfied by the boundary conditions, such as differentiability, etc. The theorem is a natural extension to partial-differential equations of the familiar result that the solution to an inhomogeneous differential equation consists of a solution to the homogeneous equation which satisfies the same initial conditions plus any particular integral of the inhomogeneous equation satisfying homogeneous initial conditions. The theorem is restricted to linear equations.

To apply this theorem to the problem under discussion several methods are possible: (1) the application of the Lagrangian multipliers

method, (2) the eigenfunction expansion method in which the projection of the boundary conditions on the complete set of eigenfunctions appropriate to the linear homogeneous operator equivalent to (100,101) is used to augment the operator, (3) the use of symbolic functions to include the boundary conditions.

Method (3) is the easiest method to apply in the present case. The preliminary details of the method proceed by finding the adjoint operator A* for the system assuming homogeneous boundary conditions; the scalar product

$$\langle A^* \underline{u}, A_e \rangle = \langle A^*, A_e \underline{u} \rangle \tag{129}$$

is formed where A_e is an extended linear operator and the domain of $\underline{u}$ now includes the non-homogeneous boundary values. The forced equality in (129) generally requires that A_e involve singularity functions; in the continuous differential equation case usually the Dirac function and its derivatives are involved. In the case of discrete systems the Kronecker function

$$\begin{aligned} \delta_{k,\ell} &= 1 \quad , \quad \ell = k \\ &= 0 \quad , \quad \ell \neq k \end{aligned} \tag{130}$$

replaces the Dirac function.

For the problem under consideration here let the solution at time state n=o be denoted as $\underline{u}^o$,

$$\begin{aligned} \underline{u}^o &= \underline{u}^o(m\,\Delta x) \\ &= \begin{pmatrix} p^o(m\,\Delta x) \\ v^o(m\,\Delta x) \end{pmatrix}. \end{aligned} \tag{131}$$

$\underline{u}^o$ is assumed to consist of two parts: a homogeneous part $\underline{u}_h^o$ which is

the solution to the linear difference equations (100,101) with

$$\underline{u}_h^o(m\Delta x) = \underline{u}^o(m) \quad , \quad m \neq o \text{ or } M-1$$
$$= o \quad , \quad m=o \text{ or } M-1, \tag{132}$$

and an inhomogeneous part $\underline{u}_i^o$ which satisfies

$$\underline{u}_i^o(m\Delta x) = o \quad , \quad m \neq o \text{ or } M-1,$$
$$= \underline{u}_i^o(o) \quad , \quad m = o,$$
$$= u_i^o(M-1) \quad , \quad m = M-1 \quad . \tag{133}$$

The values $\underline{u}_i^o(o)$, $\underline{u}_i^o(M-1)$ are the boundary values for the problem and are assumed known a priori. In terms of equations (100,101) the predicted value $\underline{u}_i^1$ has component p_m^1, v_m^1 satisfying

$$p_m^1 - p_m^o = \frac{c\Delta t}{4\Delta x}(v_{m+1}^1 - v_{m-1}^1 + v_{m+1}^o - v_{m-1}^o) + \delta_{m,o}p_o^o + \delta_{m,M-1}p_{M-1}^o \tag{134}$$

$$v_m^1 - v_m^o = \frac{c\Delta t}{4\Delta x}(p_{m+1}^1 - p_{m-1}^1 + p_{m+1}^o - p_{m-1}^o) + \delta_{m,o}v_o^o + \delta_{m,M-1}v_{M-1}^o \quad . \tag{135}$$

For the applications which require a linear combination boundary condition such as for example

$$p_o^o + b v_o^o = g^o \quad , \tag{136}$$

where b is a constant - the matched boundary condition is an example - g^o replaces p^o in (134).

Now denoting the FFT of $\underline{u}_i^o$ as $\hat{\underline{u}}_i^o$ and of $\underline{u}_i^1$ as $\hat{\underline{u}}_i^1$ and applying the transform to (134,135) results in the matrix equation.

$$G_1 \hat{\underline{u}}_i^1 = G_o \hat{\underline{u}}_i^o + \underline{u}_i^o(o) + e^{\frac{-j\,2\pi(M-1)}{M}k}\,\underline{u}_i^o(M-1) \quad . \tag{137}$$

By hypothesis $\underline{u}_i^o$ is zero for all m in (134,135) as far as the terms involved in $G_o \hat{\underline{u}}_i^o$ are concerned so that

$$G_o \hat{\underline{u}}_i^o = o \tag{138}$$

and hence

$$\hat{\underline{u}}_i^1 = G_1^{-1} \left\{ \underline{u}_i^o(o) + e^{\frac{-j2\pi(M-1)}{M} k} \underline{u}_i^o(M-1) \right\} \tag{139}$$

is the solution for the FFT spectrum of the inhomogeneous solution $\hat{\underline{u}}_i^1$ at time state n=1 Δt. The solution for $\hat{\underline{u}}_h^1$ is that already found in (122)

$$\begin{aligned} \hat{\underline{u}}_h^1 &= G_1^{-1} G_o \hat{\underline{u}}_n^o \\ &= G_{10} \hat{\underline{u}}_h^o \quad . \end{aligned} \tag{140}$$

Since the FFt is a linear mapping operation the spectrum of the sum of two functions is the sum of the individual function spectra and hence

$$\begin{aligned} \hat{\underline{u}}^1 &= \hat{\underline{u}}_h^1 + \hat{\underline{u}}_i^1 \\ &= G_{10} \hat{\underline{u}}_h^o + G_1^{-1} \hat{\underline{u}}_i^o \quad . \end{aligned} \tag{141}$$

The solution (141) represents the spectrum at time state n=1 . To proceed to time state n=2 the process is again repeated except the homogeneous spectrum at time state n=1 is now taken to be the <u>total</u> solution spectrum given in (141). The equation (141) for time state n=2 thus becomes

$$\begin{aligned} \hat{\underline{u}}^2 &= G_{10} \hat{\underline{u}}^1 + G_1^{-1} \underline{u}_i^1 \\ &= G_{10}(G_{10} \hat{\underline{u}}_h^o + G_1^{-1} \hat{\underline{u}}_i^o) + G_1^{-1} \hat{\underline{u}}_i^1 \\ &= (G_{10})^2 \hat{\underline{u}}_h^o + G_{10} G^{-1} \hat{\underline{u}}_i^o + G_1^{-1} \hat{\underline{u}}_i^1 \quad . \end{aligned} \tag{142}$$

where

$$\hat{\underline{u}}_i^1 = G_1^{-1} \left\{ \underline{u}_i^1(o) + e^{-j\frac{2\pi(M-1)}{M}k} \underline{u}_i^1(M-1) \right\} \tag{143}$$

By induction the general solution at time state $m\Delta t$ is given by

$$\hat{\underline{u}}^r = (G_{10})^n \hat{\underline{u}}_h^o + (G_{10})^{n-1} G_1^{-1} \hat{\underline{u}}_i^o + (G_{10})^{n-2} G_1^{-1} \hat{\underline{u}}_i^1 + \ldots . G_1^{-1} \hat{\underline{u}}_i^{n-1} . \tag{144}$$

An obvious thought is that since the matrix G_{10} is unitary and bounded and hence the homogeneous solution is stable independent of Δt, why not set $\Delta t = t$ the desired final time and iterate the process only once? The answer to this conjecture is negative. The system of equations (134,135) has a stability criterion which no longer is unconditional, depending in a complex way on the functional behavior of the boundary values. A moments thought leads to the realization that this must be so since only the homogeneous solution is effectively implicit; the inhomogeneous solution by (139) is explicit. Hence, the overall scheme is only semi-implicit and the magnitude of the Δt allowed depends on the magnitude of the increments in the boundary conditions between time steps.

Since the rate of change of the boundary values with time is linearly related to the frequency contained in the frequency spectrum of the boundary values, and as will be shown, the magnitude of the frequency must be low to guarantee proper non-dispersive simulation relative to the Shannon rate, the increments in the boundary conditions are generally quite small. This, coupled with the fact that only the inhomogeneous portion of the solution is not unconditionally stable, leads to the qualitative speculation that significant magnitude Δt may be used.

Computer solutions of the above algorithm have verified that such is indeed the case; quite stable and accurate results were obtained for Δt's more than twice the size of that allowed by the scheme (71,72) and with reductions factors in computation time of greater than ten. Since the number of calculations for the three-dimensional problem is of the order of the cube of that for the one-dimensional problem, the use of schemes like (110,111,112,113) is only justified for problems in which the variation in the coefficients is so rapid that prohibitively small domains would be necessary in order to use the FFT scheme.

An additional advantage of the FFT scheme is that for problems of the type of primary interest in this work in which spacial partitioning is possible, the objective of integrating the wave equation is primarily to find the steady state generated values of the unknown boundaries of a parallelepiped in terms of known values over the other boundaries of the parallelepiped. (For example if the values were known on six surfaces of a parallelepiped, what are the steady state boundary values on the seventh surface?) By virtue of (126), this would correspond to inverting the transform only for a fixed two-dimensional plane so that only L M-point inverse transforms need be calculated rather than (LxN) M-point inverse transforms - a considerable saving of computational effort.

D. BAND-LIMITED PROPERTIES OF DISCRETE MODELS; SPACIAL SAMPLING DIMENSIONS

The above algorithms, whether continuous-discrete, explicit-discrete, or implicit-discrete, all require great amounts of numerical computation, the amount depending fundamentally on the spacial partition parameter

magnitudes Δx, Δy, Δz. In section A it was claimed that the element partition size Δx, in the one-dimensional case, was determined by the operating frequency whose simulation is to be studied and the sound speed c.

1. Cutoff Properties

The simple discretized wave equation (68,69) is an exact dual of the one-dimensional periodic, (in the constant coefficient case) mass-loaded line originally studied by Baden-Powell,[5] Lord Kelvin[23] and others. The propagation properties of such lines can be found by application of Floquet's theorem;[24] if the assumed solutions

$$p(m\Delta x,t) = p_o e^{j\omega t - \gamma m \Delta x} \tag{145}$$

$$v(m\Delta x,t) = v_o e^{j\omega t - \gamma m \Delta x} \tag{146}$$

are substituted into (68,69) the determinental equation for the propagation constant γ results,

$$(\sinh k\gamma \Delta x)^2 = -\frac{\omega^2}{c^2} (\Delta x)^2 , \tag{147}$$

The form of the result (147) varies slightly depending on the finite difference formulation used but all discrete schemes result in the same qualitative behavior. For lossless propagation γ must be pure imaginary

$$\gamma = j\beta , \tag{148}$$

which gives in (147)

$$\sin \beta \Delta x = \pm\sqrt{(\frac{\omega}{c} \Delta x)^2}$$

$$= \pm \frac{\omega \Delta x}{c} . \tag{149}$$

Real β solutions to (149) occur only if

$$\frac{\omega \Delta x}{c} \leq 1 \tag{150}$$

from which the cutoff frequency f_c is obtained as

$$f_c = \frac{c}{2\pi \Delta x} \quad . \tag{151}$$

Since the velocity c is an inherent property of the medium, if propagation of a monochromatic signal of frequency f is to be simulated, f should satisfy the inequality

$$f < \frac{c}{2\pi \Delta x} \quad . \tag{152}$$

Assuming a nominal velocity in sea water of 1500 meters/sec. this restricts x to be less than

$$\Delta x < \frac{239}{f} \quad . \tag{153}$$

For a frequency of 500 Hz., $\Delta x < .478$ meters, a very short distance. The Shannon spacial sampling distance would be 1.5 meters so that the cutoff frequency restriction is more severe than the sampling restriction necessary to properly simulate the partial-differential equation.

2. Dispersion Properties

Equation (149) also indicates that the propagation is dispersive, i.e., the propagation constant β,

$$\beta = \frac{1}{\Delta x} \sin^{-1} \left(\frac{\omega \Delta x}{c} \right) , \tag{154}$$

is not a linear function of frequency except for very low frequencies. In order to make the argument of the arcsine in (154) small, Δx must be restricted to

$$\frac{\omega \Delta x}{c} \sim .1 \tag{155}$$

or

$$f \sim .1 f_c \tag{156}$$

for non-dispersive simulation. Thus the restrictions on spacial increments are very severe, especially for very higher frequencies.

IV. COMPUTER RESULTS FOR THE MATCHED BOUNDARY CONDITION

A. PARTITIONING OF THE SPACIAL DOMAIN

The integration of the acoustic wave equation over large spacial domains in which the sound speed c varies with position is usually impractical to carry out on digital computers because of the huge amounts of core required for even crude finite-cell discretizations. To implement for example the algorithm (110,111,112,113) for a rectangular parallelepiped region consisting of (25 x 25 x 10) spacial integration volumes requires a minimum of 50,000 words of memory just to store the required values of the variables. The coding of such a problem requires great amounts of indexing calculations which typically would require at least an additional 10,000 words of memory for the stored instructions, and temporary work storage. These requirements, especially in view of the fact that the parallelepiped would simulate a quite small physical volume for all but very low frequencies, has made ray theory and normal-mode model approximations the only practical methods being presently employed for investigating undersea acoustic propagation.

A problem of considerable interest is the calculation of the steady-state acoustic loss over great distance for various frequency acoustic waves in seawater wherein the sound speed c varies with position. As pointed out in the introduction, at present this calculation is carried out using ray theory to establish an approximate sound trajectory and then the loss is determined by integrating an empirical loss factor per unit length

along the trajectory.

In view of the practical limitations on core total storage size for available digital computers, the more fundamental direct integration of the wave equation is practical only if problems involving large spacial domains can be partitioned into relatively small spacial domains which can occupy core storage sequentially. The macro domain of 25 x 25 x 10 micro domains, (micro domains will refer to fundamental small elements of size Δx by Δy by Δz) might constitute such a relatively small spacial domain.

B. THE INHOMOGENEOUS LINE IMPEDANCE

In one-dimensional uniform transmission line theory a finite section of line of length ℓ can simulate a portion of an infinite line provided the voltage and current at the ends of the line are constrained by the equations (see Fig. 3)

$$v(o,t) = Z_o i_o(o,t) \tag{157}$$

$$v(\ell,t) = Z_o i(\ell,t) \tag{158}$$

where Z_o is called the characteristic impedance of the transmission line and is related to the series impedance per unit length Z_s and shunt admittance per unit length Y_{sh} by the relation

$$Z_o = \sqrt{\frac{Z_s}{Y_{sh}}} \quad . \tag{158}$$

If the line is lossless and possesses only series L and shunt C elements the equations of the line are

$$C \frac{\partial v}{\partial t} = \frac{\partial i}{\partial x} \tag{159}$$

$$L \frac{\partial i}{\partial t} = - \frac{\partial V}{\partial t} \tag{160}$$

which are the complete duals of the equations (62) and (63). If steady state solutions are obtained for (159,160) by assuming

$$\begin{pmatrix} v(x,t) \\ i(x,t) \end{pmatrix} = \mathrm{Re} \left\{ \begin{pmatrix} \hat{V}(x) \\ \hat{I}(x) \end{pmatrix} e^{j\omega t} \right\} \tag{161}$$

the voltage and current phasors $\hat{V}(x)$, $\hat{I}(x)$ satisfy the relation

$$\frac{\hat{V}(x)}{\hat{I}(x)} = \mathcal{Z}(x) \tag{162}$$

where $\mathcal{Z}(x)$ is the complex impedance

$$\mathcal{Z}(x) = Z_o \frac{\mathcal{Z}(\ell) \cos(\omega\sqrt{LC}(\ell - x)) + jZ_o \sin(\omega\sqrt{LC}(\ell - x))}{Z_o \cos(\omega\sqrt{LC}(\ell - x)) + j\mathcal{Z}(\ell) \sin(\omega\sqrt{LC}(\ell - x))} \tag{163}$$

and

$$\begin{aligned} Z_o &= \sqrt{\frac{Z_s}{Y_{sh}}} \\ &= \sqrt{\frac{L}{C}} \quad . \end{aligned} \tag{164}$$

Thus the characteristic impedance is a pure real constant only if the line is lossless or in the happy circumstance where

$$\frac{R}{L_s} = \frac{G}{C_{sh}} , \tag{165}$$

in which case though Z_s and Y_{sh} are both complex their phase angles are equal and thus cancel. In (163), $\mathcal{Z}(\ell)$ is the terminating impedance of the line; if $\mathcal{Z}(\ell)$ has the value Z_o (163) gives the result

$$\mathcal{Z}(x) = Z_o \quad , \tag{166}$$

a pure constant. The general steady state solution to (159) may be written as

$$\hat{V}(x) = V^{+} e^{-x\sqrt{Z_s Y_{sh}}} + V^{-} e^{x\sqrt{Z_s Y_{sh}}} \quad ; \tag{167}$$

if the line is lossless this results in

$$\hat{V}(x) = V^{+} e^{-j\omega\sqrt{LC}(\ell - x)} + V^{-} e^{j\omega\sqrt{LC}(\ell - x)} \tag{168}$$

$$\hat{I}(x) = \frac{\hat{V}(x)}{Z(x)} \tag{169}$$

where V^{+} , V^{-} are the amplitudes of positive and negatively traveling waves respectively. Causally if a monochromatic disturbance is propagated in the positive x direction with amplitude V^{+} the coefficient V^{-} will be zero unless there is a change in the characteristic parameters of the transmission line. In such a case reflection coefficient R can be defined by

$$R(x) = \frac{V^{-} e^{j\omega\sqrt{LC}(\ell - x)}}{V^{+} d^{j\omega\sqrt{LC}(\ell - x)}}$$

$$= R(\ell)\, e^{-j2\omega\sqrt{LC}(\ell - x)} \quad , \tag{170}$$

and is a measure of the "non-match" of the line. If the parameters L and C are not constants but functions of position Shelkunoff [25] has generalized the concept of reflection coefficient for inhomogeneous transmission lines in terms of the propagation constant $\gamma(x)$ where,

$$\gamma(x) = \sqrt{Z_s(x)\, Y_{sh}(x)}$$

$$= \alpha(x) + j\beta(x) \quad . \tag{171}$$

The derivation to follow deviates slightly from Schelkunoff's approach being specifically oriented to the acoustic problem at hand. Consider the two semi-infinite half spaces separated by the plane $x=o$. The region $x < o$ is assumed to be homogeneous with constant parameters c, ρ , χ ,

the region $x > o$ has slowly varying parameters $\varkappa(x)$ and $c(x)$. Assuming steady state dependence of the form

$$\begin{pmatrix} p(x,t) \\ v(x,t) \end{pmatrix} = \mathrm{Re} \left[\begin{pmatrix} \hat{p}(x) \\ \hat{v}(x) \end{pmatrix} e^{j\omega t} \right] \tag{172}$$

in equations (62,63) reduces the partial differential equations to the ordinary differential equations

$$\frac{d\hat{P}}{dx} = - j\omega\rho\hat{V} \tag{173}$$

$$\frac{d\hat{V}}{dx} = -j\omega\frac{\hat{P}}{\varkappa} \tag{174}$$

For $x < o$ the solution to this system for $P(x)$ assuming unit incident amplitude is

$$\hat{P}(x) = e^{-\gamma x} + R(x)\, e^{\gamma x} \tag{175}$$

where $R(x)$ is the reflection coefficient. Since the reflection coefficient at any point x_o is a measure of the mismatch as it affects the behavior for $x < x_o$ of inhomogenieties occurring for $x > x_o$, (175) as pointed out by Schelkunoff, provides a plane wave reflection coefficient which holds for any x. The propagation constant γ is obtained from (173) and (174) as

$$\gamma = j\beta$$

$$= j\frac{\omega}{c(x)} \quad . \tag{176}$$

Defining the general impedance at a point as

$$\mathcal{Z}(x) = \frac{\hat{P}(x)}{\hat{V}(x)} \tag{177}$$

and using (173), the result may be written as

$$Z(x) = \frac{\hat{P}}{-\frac{1}{j\omega\rho}\left(\frac{d\hat{P}}{dx}\right)}$$

$$= \frac{-j\omega\rho}{\frac{d}{dx}(\log \hat{P})} \quad . \tag{178}$$

From (175)

$$Z(x) = -j\omega\rho \frac{\hat{P}}{\left(\frac{d\hat{P}}{dx}\right)}$$

$$= \frac{-j\omega\rho}{-\gamma\left(e^{-\gamma x} - R(x)e^{\gamma x}\right) + R'(x)\, e^{\gamma x}}$$

$$\simeq \frac{j\omega\rho}{\gamma}\left(\frac{e^{-\gamma x} + R(x)\, e^{\gamma x}}{e^{-\gamma x} - R(x)e^{\gamma x}}\right) \tag{179}$$

if $R(x) << 1$ and slowly varying. By virtue of the continuity of the ratio $\hat{P}/\hat{V}$ the function (179) is continuous across an interface boundary even if $\varkappa(x)$ and $c(x)$ are discontinuous. By virtue of the fact that

$$\frac{d\hat{P}}{dx} = \hat{P}\, \frac{d}{dx}(\log \hat{P}), \tag{180}$$

the equation

$$\frac{d^2\hat{P}}{dx^2} - \gamma^2\hat{P} = 0 \tag{181}$$

may be written as

$$0 = \frac{d}{dx}\left(\frac{d\hat{P}}{dx}\right) - \gamma^2\hat{P}$$

$$= \frac{d}{dx}\left(\hat{P}\, \frac{d}{dx}(\log \hat{P})\right) - \gamma^2\hat{P}$$

$$= \frac{d\hat{P}}{dx}\, \frac{d}{dx}(\log \hat{P}) + \hat{P}\, \frac{d^2}{dx^2}(\log \hat{P}) - \gamma^2\hat{P}$$

$$= \hat{P} \left(\frac{d}{dx} (\log \hat{P}) \right)^2 + \hat{P} \frac{d^2}{dx^2} (\log \hat{P}) - \gamma^2 \hat{P} \tag{182}$$

or

$$\frac{d^2}{dx^2} (\log \hat{P}) = \gamma^2 - \left(\frac{d}{dx} (\log \hat{P}) \right)^2 . \tag{183}$$

By use of (178), equation (183) becomes

$$\frac{d}{dx} \left(\frac{-j\omega\rho}{Z(x)} \right) = \gamma^2 \left(\frac{-j\omega\rho}{Z(x)} \right), \tag{184}$$

or

$$\frac{dZ}{dx} + \frac{\gamma^2 Z^2}{j\omega\rho} + j\omega\rho = o . \tag{185}$$

This is a first-order non-linear equation for the impedance as a function of the propagation constant $\gamma(x)$. Alternately, (179) may be inverted to $R(x)$ in terms of $Z(x)$ and the result substituted in (183) to obtain the differential equation for the reflection coefficient

$$\frac{dR}{dx} - \frac{1}{2}(1-R^2) \frac{d}{dx} (\log (-j\gamma)) - 2\gamma R = o , \tag{186}$$

a first-order non-linear equation of the Riccati type. If

$$\frac{1}{\gamma} \frac{d}{dx} (\log (-j\gamma)) << 1 \tag{187}$$

then (186) becomes

$$\frac{dR}{dx} \simeq - 2\gamma R \tag{188}$$

which has the solution

$$R(x) = R(o)\, e^{-2\int_o^x \gamma(\xi)\, d\xi} . \tag{189}$$

By virtue of (176) it can be seen that the effect in this case is simply a phase shift due to the length of path and to first order if $c(x)$ is smoothly

varying, (in a multi-dimensional problem) to cause a bending in the direction of the wave.[1]

C. THE MATCHED BOUNDARY-CONDITION DOMAIN

The matched termination or characteristic impedance has been extensively discussed in the excellent book by Brillouin[5]; he considers the general periodic structure lattices consisting of polyatomic molecules and derives an expression giving the characteristic impedance as a function of the periodic cell internal energy. He concludes his discussion by emphasizing the difficulty involved in theoretically defining the characteristic impedance for structures in which more than one dimension is involved or where more than nearest neighbor interactions are considered.

Here the objective was to partition variable-parameter, large domains in three-dimensions with the objective of computing steady state trajectories and pressure distributions. Consider Fig. 4 which shows a large domain consisting of 27 macro-domains. Each of the individual macro-domains consists of micro-domains; (L=M=5, N=3). As the simplest and most straight-forward method to apply, an Adams-Moulton predictor-corrector, with dynamic error control program called RKAM (see program listings), was used to integrate the explicit three-dimensional finite difference scheme equivalent to the one-dimensional scheme shown in (68,69). The procedure for integrating the large domain was to integrate the problem two ways and then to compare the results:

(A) Integrate all (3+1)(5)(5)(3)(27) = 8100 equations in an overlay program until total steady state energy was attained for the entire large domain with characteristic boundary conditions

being employed on the outer-most boundaries. The energy source was placed asymmetrically in the interior of the innermost macro-domain in order to preclude any possible cancellation effects due to symmetry.

(B) Integrate the 300 equations for the innermost macro-domain with the energy source located in the same position as (1) above. Characteristic boundary conditions were again employed except their location was the boundary of the inner-most macro-domain. Upon reaching steady state conditions the FFT was employed to find equivalent oscillators for the pressure and three velocity components at each micro-domain on the surface of the inner-most macro-domain. These oscillators are then assumed to be the sources for the nearest neighbor adjacent micro-domains in the adjoining macro-domain, (the boundary micro-domains are considered to belong to both adjoining macro-domains.) In general, only the nearest neighbor macro-domains are considered in establishing steady state values for the boundaries. Since the medium is linear, the rms magnitude of the states on the boundaries can be found as the linear superposition of the quadratic contributions to the rms value of each adjacent macro-domain boundary effect. Proceeding macro-domain by macro-domain, the equivalent rms boundary values for the large domain were obtained.

The above procedures were performed on the Fleet Numerical Weather Facility CDC 6500 computer which carries out floating-point calculations to 14 decimal digits. In the case of constant coefficient domains the above two procedures produced results in excellent agreement considering the prodigous amounts of computation involved; the two computations were in agreement at all but a very small set of values to three decimal digits, There were no major discrepancies.

In the variable-parameter domain case the concept of matched characteristic impedance looses significance unless more than nearest neighbor interactions are considered because the characteristic impedance can be fundamentally defined only for periodic domains. However, when the variations in the velocity did not vary over one percent between adjoining

macro-domains the agreement of the outer boundary values was still better than one percent.

An approximate procedure to take into account the variations in c immediately beyond the boundaries of a macro-domain was investigated. This involved using the solution to the differential equation (185) to improve the value of $\hat{Z}(x)$ to be applied at the boundary micro-domain. Since only the nearest neighbor micro-domains should effect the local impedance on the boundary, the impedance at the micro-domain two micro-domains into the adjacent macro-domain normal to the boundary was assumed to have the value

$$\hat{Z}(2\ \Delta x) = \sqrt{\rho^{\kappa\,(2\,\Delta x)}} \quad . \tag{190}$$

Equation (185) was then numerically integrated back to the boundary to obtain $\hat{Z}(o)$, (the improved value to use for the ratio of $\hat{P}\,(bnd)/\hat{V}\,(bnd)$). With this added refinement the boundary values again were in agreement to three significant figures for variations in c of as much as five percent between adjacent macro-domains. Variations in c as large as five percent seldom occur over macro-domains as small as considered.

1. Stability Of The Internal Energy Criterion

The criterion used for steady state determination in a macro-domain was the total amount of internal energy contained in the macro-domain. The kinetic energy of a micro-domain is given by (at any time state r),

$$T_{micro} = \frac{1}{2}\rho\Delta x\ \Delta y\ \Delta z\ \left[v_x^2\ (\ell,m,n,r) + v_y^2(\ell,m,n,r) + v_z^2(\ell,m,n,r)\right] \tag{191}$$

and the potential energy is

$$V_{micro}(\ell,m,n,r) = \frac{\Delta x \Delta y \Delta z}{2\varkappa(\ell,m,n)} \; p^2(\ell,m,n,r) \quad . \tag{192}$$

Hence the macro-domain has the energy

$$U_{macro} = \sum_{\ell,m,n} (T_{micro} + V_{micro}) \tag{193}$$

For very low frequencies the function U_{macro} varies instantaneously but periodically with the time at twice the frequency of lowest harmonic present in the medium. Thus the criterion for steady state used was to monitor the instantaneous stored energy every period of the lowest frequency being simulated. This value was found to be stable to five decimal places.

2. Transient Behavior Of the Models

The behavior of the various integration schemes to a step input is illustrated by the simple model (72,73). In Appendix B is shown the step input pressure spacial distribution response of a 100 element one-dimensional domain of unit length ($\Delta x=0.01$). The responses are shown for t=0.,0.2,0.4,0.6,0.8,1.0,1.2,1.4, and 6 seconds. The speed c was taken to be unity. The symmetric form of the equations, (20,21) were programmed so that the characteristic impedance was unity. The results show the impulse excited transient oscillations characteristics of such models; however the matched condition is seen to produce no distinguishable reflections and predicted velocity of unit/sec is quite accurately simulated. The internal energy reached steady state to five decimal places in three seconds.

In Appendix C is shown the result of applying a unit amplitude cosine wave of frequency 1 Hz to the same line as in Appendix B. Again the characteristic transient oscillations occur along with some minor waveform distortion in the region of the maxima and minima of the cosine wave. These oscillations are caused by the shock excitation of the micro-domain self resonant frequency. The waveform distortions persist for a considerable period of time even after the energy has stabilized on the model. This emphasizes that caution is necessary in deciding when a model has reached steady state. In the problem shown an FFT of the spacial distribution could be used to indicate that only the desired simulation frequency is present but for three dimensional problems such a procedure is probably computationally too extravagant.

3. Two Dimensional Model

An example of a two-dimensional model integration with variable velocity c as a faction of one of the dimensions is shown in Appendix D. The program used was WV2525, (see program listing). An initial condition of pressure equal to unity for x=o and y varying from o to 25 Δy was applied. The velocity for y=25 Δy was three times the velocity at y=o. For stability the integration step size required was very small, $\Delta t=10^{-4}$ sec. The two-dimensional continuous-discrete system equivalent to (68,69) was used.

In Appendix E the small amplitude transient response of the plucked membrane problem is shown. This is an example of a non-zero initial condition natural response simulation.

V. APPROXIMATE FAST FOURIER TRANSFORM COEFFICIENTS

A. INTRODUCTION AND DEFINITION OF THE TRANSFORM

The Fast-Fourier-Transform (FFT) algorithm provides an efficient digital computer algorithm for performing the one-to-one discrete sample space mapping,

$$G(k) = \frac{1}{N} \sum_{n=o}^{N-1} g(n\Delta t)\, e^{-j \frac{2\pi (n\Delta t)}{N\Delta t} k} \tag{194}$$

$$g(n\Delta t) = \sum_{k=o}^{N-1} G(k)\, e^{j \frac{2\pi k}{N\Delta t} (n\Delta t)} , \tag{195}$$

where $g(n\Delta t)$, $G(k)$ are the object and image sample spaces and Δt is the sampling interval. The properties of these transform pairs have been extensively discussed in the literature.[19,26,27] For monochromatic signal sequences $g(n\Delta t)$, the spectral function $G(k)$ can be guessed or arrived at by heuristic reasoning but for general functions the summation in (194) can usually only be carried out analytically provided the particular finite series have known sum forms.

The continuous function Fourier series coefficients C_n where

$$g(t) = \sum_{n=-\infty}^{\infty} C_n\, e^{j \frac{2\pi n}{T} t} \tag{196}$$

can be obtained either by integration which is the continuous process analogous to the sum (194) or by means of symbolic differentiation[6] using impulse functions.[28] The possibility of using the symbolic method - modified for use on discrete sample spaces - is the chief result of this chapter.

B. REQUIREMENTS TO OBTAIN COEFFICIENTS BY THE IMPULSE METHOD

In the case of continuous functions the requirements for applicability of the impulse method of finding Fourier series coefficients are:

(A) A spectral representation for the Dirac function $\delta(t)$.

(B) An analytic expression relating the derivative Fourier coefficients to the original function coefficients.

(C) A piece-wise analytic expression for the function to be represented.

(D) A theorem giving the derivative for discontinuous functions.

It will now be shown that when requirements analogous to the above are satisfied for discrete functions, the FFT coefficients G(k) can be obtained.

C. SPECTRAL REPRESENTATION FOR THE SYMBOLIC IMPULSE FUNCTION

The symbolic impulse function for discrete sample spaces is the Kronecker delta; it is defined here as the mapping function $\delta_N(s-\ell)$

$$\sum_{s=o}^{N-1} \delta_N(s-\ell)\, f(s) = f(\ell) \tag{197}$$

where f(s) is a suitable testing function defined on the discrete set of points $s=o,\ldots,N-1$ and is periodic modulo N; i.e.,

$$f(s) = f(s \pm N) \quad , \quad \text{all } s \quad . \tag{198}$$

This definition is equivalent to the usual definition of the Kronecker delta as

$$\delta_N(s-\ell) = 1 \quad , \quad s = \ell,$$

$$= o \quad , \quad s \neq \ell \quad . \tag{199}$$

The function possesses the FFT coefficients

$$G(k) = \frac{1}{N} e^{-j\frac{2\pi k}{N}\ell} \tag{200}$$

as can be seen by direct application of (197) and (194). Hence

$$\delta_N(s-\ell) = \sum_{k=o}^{N-1} e^{j\frac{2\pi k}{N}(s-\ell)} \quad . \tag{201}$$

D. DEFINITION OF THE DISCRETE DERIVATIVE AND ITS SPECTRAL REPRESENTATION

Many choices are possible for defining the discrete derivative of a discrete function; the one used here is the backwards derivative commonly defined in the literature, [9]

$$g'(n\Delta t) = \frac{g(n\Delta t) - g((n-1)\Delta t)}{\Delta t} \quad . \tag{202}$$

It is a linear definition possesing the inverse

$$g(n\Delta t) = g((n-1)\Delta t) + g'(n\Delta t)\Delta t \quad , \tag{203}$$

or predictor

$$g((n+\ell)\Delta t) = g((n-1)\Delta t) + \sum_{s=n}^{n+\ell} g'(s\Delta t)\Delta t, \tag{204}$$

where $o \leq n \leq N-1$. Using this definition in conjunction with (195) results in the derivative spectral representation

$$g'(n\Delta t) = \sum_{k=o}^{N-1} G(k) \frac{\left\{1-e^{-j\frac{2\pi k}{N}}\right\}}{\Delta t} e^{j\frac{2\pi k}{N\Delta t}(n\Delta t)} \tag{205}$$

or in general for the mth derivative

$$g^{(m)}(n\Delta t) = \sum_{k=o}^{N-1} G(k) \left\{\frac{1-e^{-j\frac{2\pi k}{N}}}{\Delta t}\right\}^m e^{j\frac{2\pi k}{N\Delta t}(n\Delta t)} \tag{206}$$

An important result is obtained by applying (206) to (200)

$$\delta_N^{(m)}(s-\ell)\,\Delta t) = \sum_{k=o}^{N-1} \left\{ \frac{1-e^{-j\frac{2\pi k}{N}}}{\Delta t} \right\}^m e^{j\frac{2\pi k}{N\Delta t}(s-\ell)\Delta t} \quad . \qquad (207)$$

E. ANALYTIC ENVELOPE EXPRESSIONS FOR DISCRETE FUNCTIONS

An analytic-envelope representation for $g(n\Delta t)$ is necessary in order to be able to carry out discrete derivatives according to definition (202). Typical examples of such representation are

(A) $g_1(n\Delta t) = n\Delta t \quad , \quad o \le n \le N-1$

(B) $g_2(n\Delta t) = (n\Delta t)^2 \quad , \quad o \le n \le N-1$

(C) $g_3(n\Delta t) = n\Delta t \quad , \quad o \le n \le (N/2) - 1$

$= - n\Delta t. \quad , \quad N/2 \le n \le N-1$

(D) $g_4(n\Delta t) = \sin\left(\frac{2\pi}{N\Delta t}(n\Delta t)\right) \quad , \quad o \le N-1.$

The functions (A) and (B) are polynomial type functions with no "discontinuities in analytic-evelope" over the entire interval $o \le n \le N-1$. (C) is an example of a function described by two different envelope representations within the interval $o \le n \le N-1$; it is said to have a discontinuity in analytic envelope in the region of $n = N/2$. (D) is an example of a trancendental function representation.

F. THEOREM ON DISCRETE DERIVATIVES OF DISCONTINUOUS ANALYTIC ENVELOPE FUNCTIONS

Analogously to continuous function symbolic theory, the unit step function $Us_N(s-\ell)$, of a periodic sequence of discrete sample points s=o, 1, ..., N-1 is defined as the linear functional mapping

$$\sum_{s=o}^{N-1} Us_N(s-\ell)\, f(s) = \sum_{s=\ell}^{N-1} f(s) \quad . \tag{208}$$

From (194,195) it is clear that the results for the coefficients are independent of Δt; it thus will be dispensed with in subsequent analyses.

$Us_N(s-\ell)$ as defined in (208) is equivalent to the piece-wise definition

$$\begin{aligned} Us_N(s-\ell) &= 1 \quad , \qquad s \geq \ell \\ &= o \quad , \qquad s < \ell \quad . \end{aligned} \tag{209}$$

This function possesses the property that

$$\begin{aligned} \sum_{s=\ell}^{N-1} Us'_N(s-\ell)\, f(s) &= \sum_{s=o}^{N-1} \{Us_N(s-\ell) - Us_N(s-1-\ell)\}\, f(s) \\ &= \left\{ \sum_{s=\ell}^{N-1} - \sum_{s=\ell+1}^{N-1} \right\} f(a) \\ &= f(\ell) \\ &= \sum_{s=o}^{N-1} \delta_N(s-\ell) f(s) \quad , \end{aligned} \tag{210}$$

so that symbolically

$$Us'_N(s-\ell) = \delta_N(s-\ell) \tag{211}$$

A change in a continuous analytic-envelope function can thus be described by

$$g(n) = g_1(n) + \alpha\, Us_N(n-\ell) \tag{212}$$

where $g_1(n)$ has the same analytic-envelope representation for all n. This represents the situation of a discontinuity in analytic -envelope at the point $n=\ell$. See Fig. 5. The discontinuity is a jump of magnitude α. If (211) is used in (212) the result for the discrete derivative is

$$g'(n) = g_1'(n) + \alpha \delta_N(n-\ell) \quad . \tag{213}$$

This proves the theorem:

Theorem: The discrete derivative of a discrete function possessing a discontinuity in analytic-envelope at the point $n=\ell$, is the backwards derivative defined in (202) for all $n \neq \ell$ plus a Kronecker delta function at $n=\ell$ of amplitude α, where

$$\alpha = g(\ell) - g_1(\ell-1) \, . \tag{214}$$

G. EXAMPLES

1. Coefficients of a Sawtooth Waveform

The preceding results and theorem can now be applied to a simple problem to illustrate the method. Consider the periodic sawtooth waveform shown in Fig. 6 (a). The function has the representation

$$g(n) = n \quad , \quad o \leq n \leq N-1$$

$$g(n \pm N) = g(n) \quad . \tag{215}$$

Assuming $g(n)$ has an FFT then

$$g(n) = \sum_{k=o}^{N-1} G(k) e^{j\frac{2\pi k}{N} n}$$

$$= n \, , \quad , \quad o \leq n \leq N-1. \tag{216}$$

If (210) is formally differentiated by means of (196) there results, (see Fig. 6 (b))

$$g'(n) = n - (n-1)$$

$$= 1 \quad , \quad 1 \leq n \leq N-1 \tag{217}$$

and

$$g'(n) = o - (N-1)$$

$$= -(N-1) \quad , \quad n=o \ . \tag{218}$$

The results (211) and (212) can be combined into the single result

$$g'(n) = 1 - N\,\delta_N(n) \quad , \quad o \le n \le N-1 \tag{219}$$

where now the "1" in (219) is again a continuous-envelope discrete function holding for <u>all n</u>, $o \le n \le N-1$. In general, in applying the method, since continued differentiations of a discrete function tend to introduce more and more discontinuities in the higher order derivatives, adjustments such as performed in (219) greatly reduce the "book-keeping" required.

If now (219) is differentiated a second time the result is simply

$$g''(n) = o - N\,\delta'_N(n) \quad , \quad o \le n \le N-1 \ . \tag{220}$$

Using (206) and (207) gives for (220) in terms of FFT expansions

$$\sum_{k=o}^{N-1} G(k) \left\{ 1 - e^{-j\frac{2\pi k}{N}} \right\} 2 \; e^{j\frac{2\pi k}{N} n}$$

$$= \frac{1}{N} \sum_{k=o}^{N-1} \left[-N\left(1 - e^{-j\frac{2\pi k}{N}}\right) \right] e^{j\frac{2\pi k}{N} n} \ . \tag{221}$$

(221) must be true for arbitrary n which requires that the coefficients of $\exp(2\pi kn/N)$ vanish for all $k \neq o$; hence

$$G(k) = -\frac{1}{\left\{ 1 - e^{-j\frac{2\pi k}{N}} \right\}} \quad , \quad k = 1,2,3, \ldots, N-1 \ . \tag{222}$$

The coefficient for $k = o$ cannot be obtained from (221) since the

bracketed term vanishes; this is to be expected since k=o represents the constant term of the expansion and this must vanish upon differentiation. The k=o term can be however, trivially obtained as the mean value of the function over the interval,

i.e., from (194) with k=o,

$$G(o) = \frac{1}{N} \sum_{n=o}^{N-1} g(n) \quad . \tag{223}$$

The results given in (222) and (223) have been checked using several FFT[29] programs to at least six decimal places.

The procedure for the use of the method is now clear; if the discrete analytic-envelope function is polynomial in form it is evident that continued application of (202) must eventually result in the vanishing of the discrete analytic-envelope portion of the function leaving only a series of Kronecker deltas of various orders whose FFT coefficients can be written down by inspection.

2. Coefficient of Recursive Waveforms -- The Exponential Function

The question of how to treat infinite expansion polynomial analytic-envelope functions is also of interest. This would include functions such as the sine, exponential, etc.

As an example of the the treatment of these functions consider the problem of finding G(k) when

$$g(n) = e^{-na} \quad , \qquad o \leq n \leq N-1 \tag{224}$$

g(n) being periodic (see Fig. 7)

$$g(n \pm N) = g(n) \, . \tag{225}$$

Applying the derivative theorem gives

$$g'(n) = e^{-na} - e^{-(n-1)a} \quad , \quad 1 \leq n \leq N-1$$

$$= (1-e^{-(N-1)a})\, \delta_N(n) \quad , \quad n = o \, . \tag{226}$$

This can be expressed for all n in the interval as

$$g'(n) = e^{-na} - e^{-(n-1)a} + (1-e^{-(N-1)a})\, \delta_N(n) - (1-e^{a})\, \delta_N(n)$$

$$= e^{-na}(1-e^{a}) + (e^{a}-e^{-(N-1)a})\, \delta_N(n) \quad , \tag{227}$$

or

$$g'(n) = g(n)\,(1-e^{a}) + e^{a}(1-e^{-Na})\, \delta_N(n) \quad , \quad o \leq n \leq N-1 \, . \tag{228}$$

(228) is typical of the behavior of non-finite expansion polynomials for continuous functions: the derivative of such functions can be written in terms of the original function or certain linear combinations of the previous derivatives [28] The analogous result can be verified to hold true for discrete functions to first order; i.e., since the derivative in (202) includes only linear difference terms and ignores in (203) contributions to $g(n\,\Delta t)$ proportional to $(\Delta t)^2$, $(\Delta t)^3$, etc., the general expansion for the m-th derivative $g^{(m)}(n)$

$$g^{(m)}(n) = h_o g(n) + h_1 g^{(1)}(n) + \ldots h_{m-1}\, g^{(m-1)}(n), \tag{229}$$

where h_o , h_1 , .. h_{m-1} , are constants will deviate seriously from the analogous expansion in the continuous case unles Δt is small. However, though the form of the expansion (229) is different from that for continuous functions the result (229) is completely correct for discrete functions if the constant terms are correctly taken into account. Such functions are

thus linearly recursive. If (228) is now expressed in terms of the FFT the result is

$$\sum_{k=o}^{N-1} \left\{ (1-e^{-j\frac{2\pi k}{N}}) - (1-e^{a}) \right\} G(k)\, e^{j\frac{2\pi k}{N} n}$$

$$= \frac{1}{N} \sum_{k=o}^{N-1} \left\{ e^{a}(1-e^{-Na}) \right\} e^{j\frac{2\pi k}{N} n} \quad . \tag{230}$$

This requires that

$$G(k) = \frac{e^{a}(1-e^{-Na})}{N(e^{a}-e^{j\frac{2\pi k}{N}})}$$

$$= \frac{1-e^{-Na}}{N(1-e^{-(a+j\frac{2\pi k}{N})})} \quad , \quad k=1,2,3,\ldots,N-1 \; . \tag{231}$$

This particular result can be checked exactly analytically, for the function (224), when substituted into (194) becomes

$$G(k) = \frac{1}{N} \sum_{n=o}^{N-1} e^{-na} e^{-j\frac{2\pi k}{N} n}$$

$$= \frac{1}{N} \sum_{n=o}^{N-1} e^{-n(a+j\frac{2\pi k}{N})}$$

$$= \frac{1}{N} \sum_{n=o}^{N-1} x^{n} \quad , \tag{232}$$

where

$$x = e^{-(a+j\frac{2\pi k}{N})} \quad . \tag{233}$$

Equation (232) is simply a finite geometric series with the sum

$$G(k) = \frac{1}{N} \frac{1-x^{(N-1)+1}}{(1-x)}$$

$$= \frac{1}{N} \left(\frac{1-x^{N}}{1-x} \right)$$

$$= \frac{1-e^{-Na}}{N\left(1-e^{-\left(a+j\frac{2\pi k}{N}\right)}\right)} \quad , \quad k=o,1, \ldots, N-1 \qquad (234)$$

Note that in this case the term k=o has been obtained along with the usual other terms; this is because in (22) the derivative has been expressed recursively in terms of the complete original function e^{-na}.

F. GENERAL PROCEDURE

The above two examples illustrate the general procedure to be followed for finding the coefficients of a Fast Fourier Transform. The procedure may be summarized and generalized as follows:

(A) If the analytic-envelope is finite polynomial in form continually differentiate the function until only Kronecker delta functions remain. After each differentiation, adjust the analytic form so that the particular derivative holds at all points within the interval by extending the analytic-envelope over the entire domain $o \leq n \leq N-1$ with suitable modification of the Kronecker delta amplitude.

(B) Equate the spectral representations for g(n) to obtain the coefficients for all $k \neq o$ terms. The k=o term is obtained as the average value of the discrete waveform. This can be obtained in many cases exactly by use of the finite arithmetic or geometric sum formulas.

(C) If the analytic-envelope consists of sub-regions over the interval $o \leq n \leq N-1$ defined by two or more different functions $g_1(n)$, $g_2(n)$, etc., the problem can be handled by multiplying the appropriate sub-region function definitions by unit step functions for the appropriate regions. Differentiation of the resulting products using the usual product rule will lead to a simplified analytic-envelope function plus the necessary Kronecker deltas. The necessary work quickly becomes prohibitive unless the simplified analytic-envelope can be easily

extended over the whole interval as done in the previous problems.

(D) If the function is transcendental, use the recursive procedure shown in the second example.

In conclusion, the method presented in this section provides an analytic method for determining closed form formulas for FFT coefficients. These expressions can be used in theoretical discussions to study the behavior of the coefficients for various sampling intervals, size of discontinuity present, accuracy of computer programs, etc.

VI. TWO APPROACHES TO THE OPTIMUM ACOUSTIC ARRAY PROCESSOR

A. INTRODUCTION

A subject of great importance and deserving of continued study is the detection of acoustic communication signals in the presence of a background noise field.[30] In this chapter several approaches to the solution of this problem are reviewed and the results of simulating one of the approaches on a computer are reported and discussed.

The usual means of detecting and receiving acoustic signals, corrupted by noise is by a spacially-distributed array of acoustic sensors. Such arrays serve the fundamental dual purposes of providing multiplication of individual sensor gain and spacial correlation. A typical array of sensors together with the associated signal processing network is shown in Fig. 8.

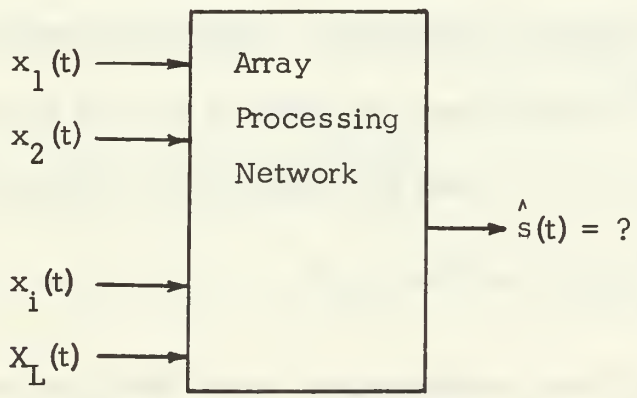


Fig. 8. Array with associated processing network.

Although analog, digital, or a combination of techniques may be involved in the signal processing network in Fig. 8, a few qualitative

generalizations can be made. If the input signal s(t) is wideband and quite high information rates are involved the decision to "go analog" is almost certain. The required computation rates for real time processing are very large in such cases even taking into account the possibility of using such tools as the FFT algorithm;thus digital processing is restricted to relatively low frequencies. However, the FFT algorithm, coupled with very high speed special purpose computers, have made possible data rates of 10 kHz and higher so that in the near future the use of digital real-time parallel processing may be expected to increase considerably.[31]

There are many different approaches to the problem of optimum array processing; two main approaches will be discussed here. The first approach discusses the possibility of modifying classical analog optimum quadratic-cost-function detection and estimation theory to make use of the FFT algorithm; the second approach considers the use of Widrow adaptive array processor and possible modifications.

B. THE MODIFIED OPTIMUM DETECTION - ESTIMATION ARRAY PROCESSOR

Following Cox[32] closely let it be assumed that the L sensors in Fig. 8 have inputs $x_i(t)$, i=1,L and that the i-th input can be written as

$$x_i(t) = \int_{-\infty}^{\infty} m_i(t-\tau)s(\tau)\,d\tau + n_i(t) \quad , \; i=1,2,\ldots,L \qquad (235)$$

where $m_i(t)$ is a known linear transformation which takes into account the i-th time delay of arrival of the scalar signal s(t). $m_i(t)$ thus provides the generality to account for local dispersive properties of the medium, non-uniform array spacing, and phase distortion of particular sensors. The

function $n_i(t)$ is assumed to be additive medium noise. The relation (235) can be re-written as the vector equation

$$\underline{x}(t) = \int_{-\infty}^{\infty} \underline{m}(t-\tau)s(\tau)\, d\tau + \underline{n}(t) \tag{236}$$

where now $\underline{x}$, $\underline{m}$, and $\underline{n}$ and L-dimensional column vectors. Several investigators have formed large hyper-vector array ensembles of the quantities $\underline{x}$, $\underline{m}$, and $\underline{n}$ in the form of correlation matrices at various times t_1, t_2, etc., in order to then use optimum estimation techniques [30,31]; these procedures without the assumption of stationarity are confined to very low frequencies because of the great magntiude of the computation requirements.[33,34]

The assumption of stationarity will be made in the following sections in order to apply frequency domain techniques. Let it be assumed that

$$\underline{x}(t) = \sum_{k=-\infty}^{\infty} \underline{\psi}'(k)\, e^{j2\pi k f_1 t} \quad , \quad s(t) = \sum_{k=-\infty}^{\infty} \sigma'(k) e^{j2\pi k f_1 t}$$

$$, \quad \underline{n}(t) = \sum_{k=-\infty}^{\infty} \underline{\eta}'(k) e^{j2\pi k f_1 t} \tag{237}$$

$$\underline{m}(t) = \int_{-\infty}^{\infty} \underline{\mu}'(f) e^{j2\pi ft} df \, , \tag{238}$$

where

$$f_1 = \frac{1}{T} \quad . \tag{239}$$

Then, substituting in (236) requires that

$$\underline{o} = \sum_{k=-\infty}^{\infty} \left\{ (\underline{\psi}'(k) - \underline{\eta}'(k)) e^{j2\pi k f_1 t} - \int_{-\infty}^{\infty} \underline{\mu}'(f)\sigma'(k) e^{j2\pi ft} \right.$$

$$\left. \cdot \int_{-\infty}^{\infty} d\tau\, e^{j2\pi\tau(kf_1 - f)} \right\}$$

$$= \sum_{k=1}^{\infty} \left\{ (\underline{\psi}'(k) - \underline{\eta}'(k)) e^{j2\pi kf} - \int_{-\infty}^{\infty} \underline{\mu}'(f)\sigma'(k) e^{j2\pi ft} \delta(f - f_1 k)\, df \right\}$$

$$= \sum_{k=\infty}^{\infty} \left\{ \underline{\psi}'(k) - \underline{\mu}'(k)\,\sigma'(k) - \underline{\eta}'(k) \right\} e^{j2\pi f_1 kt} \quad . \tag{240}$$

This can vanish identically for arbitrary t if and only if the bracket term vanishes which gives

$$\underline{\psi}'(k) = \underline{\mu}'(kf_1)\,\sigma'(k) + \underline{\eta}'(k) \qquad , \qquad \text{all } k \quad . \tag{241}$$

(241) suggest arranging the equation in the matrix forms

$$\underline{\psi} = \begin{pmatrix} \underline{\psi}'(1) \\ \underline{\psi}'(2) \\ \vdots \end{pmatrix}, \quad \underline{\eta} = \begin{pmatrix} \underline{\eta}'(1) \\ \underline{\eta}'(2) \\ \vdots \end{pmatrix}, \quad \sigma = \begin{pmatrix} \sigma'(1) \\ \sigma'(2) \\ \vdots \end{pmatrix} \tag{242}$$

$$\underline{\mu} = \begin{pmatrix} \underline{\mu}'(f_1) & \underline{0} & \underline{0} & \cdots \\ \underline{0} & \underline{\mu}'(2f_1) & \underline{0} & \\ \vdots & & \underline{\mu}'(3f_1) & \ddots \end{pmatrix}$$

$$= \text{diag}\ (\underline{\mu}'(f_1), \underline{\mu}'(f_2), \ldots) \tag{243}$$

or

$$\underline{\psi} = \underline{\mu}\,\sigma + \underline{\eta} \tag{244}$$

where $\underline{\psi}'(1)$ represents the Fourier series coefficients of the entire array of L elements at frequency f_1, $\underline{\psi}'(2)$ the Fourier coefficients of the entire array at frequency $2f_1$, etc. If the signal $\underline{s}(t)$, input vector $\underline{x}(t)$ and noise $\underline{n}(t)$ are not band-limited the number of rows of the equation (244) is infinite; the usual assumption is that $\underline{x}(t)$ is band-limited to some upper frequency f_{max} where

$$f_{max} = k_{max} f_1$$

$$= \frac{k_{max}}{T} \quad , \tag{245}$$

and T is the observation period.

Denoting the noise auto-spectrum at frequency kf_1 by $\underline{R}(k)$ where

$$\underline{R}(k) = E\left\{\underline{\eta}(k)\, \underline{\eta}^*(k)\right\} \quad , \tag{264}$$

and assuming that the period of observation T is large compared to the maximum lag for which the auto-correlation function is significantly different from zero, the cross spectrum $\underline{R}(k,\ell)$ satisfies

$$\underline{R}(k,\ell) = E\left\{\underline{\eta}(k)\, \underline{\eta}^*(\ell)\right\}$$

$$= \underline{o} \quad , \qquad \text{all } k \neq \ell; \tag{247}$$

the noise covariance matrix is thus of the form

$$\underline{R} = \begin{bmatrix} \underline{R}(1) & \underline{o} & \underline{o} & \cdots \\ \underline{o} & \underline{R}(2) & \underline{o} & \\ \underline{o} & \underline{o} & \underline{R}(3) & \\ \vdots & & & \ddots \end{bmatrix}$$

$$= \text{diag}\left(\underline{R}(1), \underline{R}(2), \underline{R}(3), \ldots\right) . \tag{248}$$

In (246) $\underline{\eta}^*$ is the complex conjugate transpose of $\underline{\eta}$. The matrix $\underline{R}$ is square and each submatrix on the diagonal is of dimension LxL.

Suppose $s(t)$ is assumed random with zero mean; a spectral covariance matrix $\underline{P}$ where

$$\underline{P} = E\left\{\sigma(k)\, \sigma^*(\ell)\right\} \quad , \tag{249}$$

can be defined. Under the assumption again that the expansion period T is long, $\underline{P}$ is also diagonal having the form

$$\underline{P} = \begin{bmatrix} p_1 & o & o & \cdots \\ o & p_2 & o & \\ o & o & p_3 & \ddots \\ \vdots & & & \ddots \end{bmatrix}$$

$$= \text{diag}\,(p_1, p_2, p_3 \ldots) \quad , \tag{250}$$

where p_k is the power in the spectral line kf_1. Because of diagonal forms of the matrices $\underline{\mu}$, $\underline{P}$, and $\underline{R}$, the test statistic for the detection problem, ζ, which when compared with a threshold statistic determines whether a signal has occured or not, can be expressed in the simple form

$$\zeta = \frac{1}{2} \sum_{k=1}^{k_{max}} \left| a(k)\, (\underline{\mu}'(k))^*\, (\underline{R}(k))^{-1}\, \underline{\psi}'(k) \right|^2 \quad , \tag{251}$$

where

$$|a(k)|^2 = \frac{p_k}{1 + p_k (\underline{\mu}'(k))^* (\underline{R}(k))^{-1} \underline{\mu}'(k)} \quad . \tag{252}$$

The form (251) is simply a sum of terms over the spectrum of the signal and is derived as the ratio of the conditional probabilities

$$\zeta = \frac{p(\underline{\psi} / (\underline{\mu}\sigma + \underline{\eta}))}{p(\underline{\psi}' / \underline{\eta})} \quad , \tag{253}$$

where

$p(\underline{\psi} / (\underline{\mu}\sigma + \underline{\eta}))$ = probability that during an observation interval T the input spectra contains the signal plus noise spectra,

$p(\underline{\psi} / \underline{\eta})$ = probability that during an observation interval T the input spectra contains only noise spectra.

If now a vector spectral function $\underline{\theta}^*(k)$,

$$\underline{\theta}^*(k) = \underline{\mu}'^*(k)\, (\underline{R}(k))^{-1} \tag{254}$$

and a scalar spectral response function $\Omega(k)$,

$$\Omega(k) = a(k)\, \underline{\theta}^*(k)\, \underline{\psi}'(k), \tag{255}$$

are defined Parseval's theorem gives the simple result for $\zeta(T)$

$$\zeta(T) = \frac{1}{2T} \int_0^T \left| \sum_{-k_{max}}^{k_{max}} \Omega(k) e^{j2\pi k f_1 t} \right|^2 dt \quad . \tag{256}$$

Equation (256) in practice can be accomplished by averaging the square law detected output of the spectral function $\Omega(k)$. The function $\underline{\theta}^*(k)$ performs the operations of pre-whitening the input signal and then matching to the assumed known spacial structure of the signa. Fig. 9 shows the complete system; it is composed of a system of weighted bandpass filters on each sensor followed by spacial spectral summation, multiplication, square law detection and averaging.

This processor can be generalized to supply with but slight modification not only detection and optimum estimation of a random signal $\hat{s}(t)$ but also detection and optimal estimation for the known and unknown signal problems. Figure 10 shows the general all-purpose array processor assuming analog processing.

The use of an analog spectrum processor has several disadvantages even though high real-time data rates are possible. Particularly objectionable is the fact that the banks of narrow band-pass "comb" filters required at the outputs of each sensor are difficult to maintain in precise adjustment track between sensors and are quite difficult to shift in frequency. In addition the process of analog multiplication is rather crude.

As an alternative to the analog spectrum processor, the possibility of using the discrete FFT algorithm to perform the spectral analysis was considered. The essential modifications in the above analysis are as follows.

Consider equations (237,238). Assuming stationarity the quantities $\underline{x}(t)$, $\underline{n}(t)$ and $s(t)$ could be discretized by a digital sampler at the output of each sensor to provide the data values $\underline{x}(i\,\Delta t)$, $\underline{n}(i\,\Delta t)$, $s(i\,\Delta t)$ where $o \leq i \leq (N-1)$ and the sampling interval Δt in any observation interval T satisfies

$$N\,\Delta t = T\ .$$

Since the maximum harmonic that can be represented by the finite discrete N point Fourier transform is

$$f_{max} = (\frac{N}{2} - 1)f_1 = \frac{(\frac{N}{2} - 1)}{T}\ ,$$

the output of each sensor must be low-pass filtered to exclude all components of frequency greater than f_{max} if the finite band-width of the sensor has not already provided the needed filtering. Failure to include such filtering will cause aliasing.[26] The discrete equivalent of (237) is thus

$$\underline{x}(\xi) = \sum_{k=o}^{N-1} \underline{\psi}'(k)e^{j\frac{2\pi k}{N}\xi}\ ,\quad \underline{n}(\xi) = \sum_{k=o}^{N-1} \underline{\eta}'(k)e^{j\frac{2\pi k}{N}\xi}\ ,$$

$$(\xi) = \sum_{k=o}^{N-1} \sigma(k)e^{j\frac{2\pi k}{N}\xi}\ ,\quad \xi = o,1,\ldots,N-1. \qquad (257)$$

In (238) the spacial weighting vector $\underline{m}(t)$ is represented by a bi-lateral infinite Fourier transform and not a Fourier series. Ignoring this important difference for the moment and assuming that $\underline{m}(\xi)$ can be represented by the finite discrete transform

$$\underline{m}(\xi) = \sum_{\ell=0}^{N-1} \underline{\mu}'(\ell) e^{j\frac{2\pi\ell}{N}\xi}, \quad \xi = 0, 1, \ldots, N-1 \tag{258}$$

the discrete equation equivalent to (240) becomes

$$0 = \sum_{k=0}^{N-1} \left\{ (\underline{\psi}'(k) - \underline{\eta}'(k)) e^{j\frac{2\pi k}{N}\xi} \sum_{\tau=0}^{N-1} \sum_{\ell=0}^{N-1} \underline{\mu}'(\ell) e^{j\frac{2\pi\ell}{N}(\xi-\tau)} \cdot \sigma(k) e^{j\frac{2\pi k}{N}\tau} \right\}$$

$$= \sum_{k=0}^{N-1} \left\{ (\underline{\psi}'(k) - \underline{\eta}'(k)) e^{j\frac{2\pi k}{N}\xi} - \sum_{\ell=0}^{N-1} \underline{\mu}'(\ell) e^{j\frac{2\pi\ell}{N}\xi} \sigma'(k) \sum_{\tau=0}^{N-1} e^{j\frac{2\pi\tau}{N}(k-\ell)} \right\}$$

$$= \sum_{k=0}^{N-1} \left\{ (\underline{\psi}'(k) - \underline{\eta}'(k)) e^{j\frac{2\pi k}{N}\xi} - \sum_{\ell=0}^{N-1} \underline{\mu}'(\ell) e^{j\frac{2\pi\ell}{N}\xi} \sigma'(k)\, \delta_N(k-\ell) \right\}$$

$$= \sum_{k=o}^{N-1} \left\{ \underline{\psi}'(k) - \underline{\mu}'(k)\, \sigma'(k) - \underline{\eta}'(k) \right\} e^{j\frac{2\pi k}{N}\xi} \quad . \tag{259}$$

Employing the same argument as in (240) the result

$$\underline{\psi}'(k) = \underline{\mu}'(k)\, \sigma(k) + \underline{\eta}'(k) \qquad , \quad k=o,1, \ldots, N-1, \tag{260}$$

follows. Thus if (258) is valid the problem is again cast into the framework of (241) and the subsequent analysis follows directly.

Now equation (238) in the analog formulation must involve a continuous frequency spectrum in order that <u>any</u> possible phase shift corresponding to any possible time delay $\delta(t-t_i)$ relating the response time at the i-th sensor to that at some convenient origin may be represented. The Kronecker delta in (259) does not possess such resolution; it can represent at most discrete time delays, separated by a minimum time of Δt sec. and a maximum time of $N\,\Delta t = T$ sec. Thus only certain discrete angular resolution is possible with the L element array under the representation restriction (238); the array has an angular boresight error of

$$\delta\phi \sim \frac{c\,\Delta t}{L\,\delta\lambda} \tag{261}$$

where c is the local acoustic velocity and $\delta\lambda$ is the fraction of a wavelength between elements for a uniform array detecting acoustic signals of wavelength λ. Hence for fine angular resolution Δt should be small. A more fundamental restriction is imposed by the maximum time delay representable i.e., $N\Delta t$. Typical high resolution acoustic arrays may be of 30 λ to 50 λ in aperture and in an endfire configuration this means that time delays of the order of

$$\text{delay} \sim \frac{50\,\lambda}{c} \tag{262}$$

across the array are possible. As a consequence even if T were not required to be long in order to guarantee the diagonal properties of the matrices $\underline{P}$ and $\underline{R}$, it would be necessary for it to be long in order to satisfy (257).

The net result of these considerations is that provided T is at least greater than some criterion such as (262) and Δt is small enough to satisfy desired angular resolution requirements such as (261) the analysis in (259) is valid and hence the FFT digital spectral analysis can be applied to optimally detect and estimate in the least squares sense the presence of a signal s(t) in the presence of background noise.

C. ADAPTIVE ARRAYS

1. The Widrow Scheme

The approach outlined above is classical in nature and relies on the statistical properties of signals alone to perform detection and estimation. The procedure requires the inversion of the noise covariance matrix which involves considerable computation and can only be completely avoided if the noise is known a priori to be white. Further, apart from increased gain (which in general amplifies both the signal and noise) no particular correlation properties of the array are utilized.

Conceptually, as previously stated in the introduction , an ideal array processor should be able to adapt to both the specific spacial-location medium properties where the array is to be placed and an incoming signal.

Widrow[40] and his co-workers have devised an adaptive scheme which appears to have some of these properties as well as other advantages

over the classical procedure in those cases in which the statistics of the signal s(t) are known and the noise field has non-uniform directional properties. No particular properties of the noise field are required except that the incident noise direction be different from that of the desired incident signal. The initial development given below is considerably expanded over that included in Widrow's paper, being a generalization to the vector case of a paper by Mantey.[41]

In Fig. 11 is shown a discrete sampled-data network involving time delay networks for the i-th array element shown in Fig. 8. From Fig. 11, the output $y_i(k)$ can be written as a function of the previous inputs and outputs as

$$y_i(k) = a_1 y_i(k-1) + a_2 y(k-2) + \ldots + a_N y_i(k-N) + b_o x_i(k) + b_1 x_i(k-1) + \ldots + b_M x_i(k-M) \quad , \qquad (263)$$

where $x_i(k)$ is the input sample at time $k\Delta t$, $x_i(k-1)$ is the input sample at time $(k-1)\,\Delta t$, $y_i(k)$ is the output of the network at time $k\,\Delta t$, $y_i(k-1)$ is the output of the network at time $(k-1)\,\Delta t$, etc. The coefficients $a_1, \ldots, a_N$ are referred to as feedback coefficients and the coefficients $b_o, \ldots, b_M$ are called feedforward coefficients. The term feedback here comes from the fact that the output $y_i(k)$ is determined not only by the present and past inputs but also by the past outputs of the network. Such networks thus possess "memory" in that, indirectly, the output $y_i(k)$ depends on all previous inputs $x_i(k)$ of the network. This is a highly desirable property. In detection theory the use of the matched filters to estimate threshold levels optimally in the least mean squares sense is

important[32]; such a filter is required to have an impulse response which is the convolution of the known signal waveform which is to be detected. Thus if the network in Fig. 11 is required to simulate such a filter it is necessary that its impulse response be as close to the waveform assumed a priori to be contained in $x_i(t)$ as possible. An important consideration in this regard is the relative efficiency in providing memory of each of the terms in (263); the impulse response is non-vanishing for the purely feedforward terms $b_o, b_1, \ldots, b_M$, only over the intervals $(o \leq k\Delta t \leq M\Delta t)$ whereas the feedback coefficients provide semi-infinite non-vanishing response, $(o \leq k\Delta t \leq \infty)$. Thus a much simpler feedback network can provide theoretically a sampled impulse response which can be made more nearly a "match" for the desired signal. As pointed out by Mantey[57] the cost in computing time and not computer storage is the essential limitation on the real time application of digital signal processing schemes; it can be seen from (263) that each feedback section "costs" the same as each feed-forward section in computing time. The sample data network containing a single input $b_o x(k)$, and N feedback terms $a_N y(k-N), \ldots a_1 y(k)$ is called an auto-regressive network and has been extensively studied by Whittle[42] and Claerbout,[43] who conclude that the network has apparent advantages over other schemes in that it appears to yield more physically realizable models for stochastic processes which are time variable.

Referring to Fig. 11 and equation(263) it is clear that the output of the entire L element array can be written in vector form as

$$\underline{y}(k) = \sum_{n=1}^{N} A_n \underline{y}(k-n) + \sum_{m=o}^{M} B_m \underline{x}(k-m) \tag{264}$$

where

$$\underline{y}(k) = \begin{matrix} y_1(k) \\ \vdots \\ y_L(k) \end{matrix}, \quad \underline{x}(k) = \begin{matrix} x_1(k) \\ \vdots \\ x_L(k) \end{matrix} \tag{265}$$

$$A_n = \text{diag}(a_{1n}, a_{2n}, \ldots a_{Ln}), \; B_m = \text{diag}(b_{1m}, b_{2m}, \ldots, b_{2m}). \tag{266}$$

In (264) there are N LxL matrices A_n and M+1 LxL matrices B_m.

The theory of sample-data systems may be most elegantly treated by the Z transform[44], which is defined in terms of a complex variable z and a uniform sample sequence f(k) as

$$F(z) = \sum_{k=o}^{\infty} f(k)\, z^{-k}$$

$$= \mathcal{Z}(f(k)) \tag{267}$$

and has the property that the transform of a unit pulse delayed ℓ samples in time $\delta_{k,\ell}$, is

$$\mathcal{Z}(\delta_{k,\ell}) = z^{-\ell} . \tag{268}$$

Applying the transform to the system (264) thus gives

$$\underline{Y}(z) = \sum_{n=1}^{N} A_n \mathcal{Z}(\underline{y}(k-n)) + \sum_{m=o}^{M} B_m \mathcal{Z}(\underline{x}(k-m))$$

$$= \sum_{n=1}^{N} A_n z^{-n} \underline{Y}(z) + \sum_{m=o}^{M} B_m z^{-m} \underline{X}(z)$$

$$= \sum_{n=1}^{N} (A_n z^{-n}) \underline{Y}(z) + \sum_{m=o}^{M} (B_m z^{-m}) \underline{X}(z) , \tag{269}$$

where

$$\underline{Y}(z) = \mathcal{Z}(\underline{y}(k)) \quad , \quad \underline{X}(z) = \mathcal{Z}(\underline{x}(k)) \quad . \tag{270}$$

Defining the matrices A and B by

$$\begin{aligned}
A &= \sum_{n=1}^{N} A_n z^{-n} \\
&= \sum_{n=1}^{N} \operatorname{diag}(a_{1n}, a_{2n}, \ldots, a_{Ln}) z^{-n} \\
&= \sum_{n=1}^{N} \operatorname{diag}(a_{1n} z^{-n}, a_{2n} z^{-n}, \ldots, a_{Ln} z^{-n}) \\
&= \operatorname{diag}\left(\sum_{n=1}^{N} a_{1n} z^{-n}, \sum_{n=1}^{N} a_{2n} z^{-n}, \ldots, \sum_{n=1}^{N} a_{Ln} z^{-n}\right), \\
B &= \sum_{m=o}^{M} \operatorname{diag}(b_{1m}, b_{2m}, \ldots, b_{Lm}) z^{-m} \\
&= \operatorname{diag}\left(\sum_{m=o}^{M} b_{1m} z^{-m}, \sum_{m=o}^{M} b_{2m} z^{-m}, \ldots, \sum_{m=o}^{M} b_{Lm} z^{-m}\right),
\end{aligned} \tag{271}$$

the equation (269) becomes

$$\underline{Y}(z) = A\underline{Y}(z) + B\,\underline{X}(z), \tag{272}$$

which may be written as

$$\begin{aligned}
\underline{Y}(z) &= (I-A)^{-1} B\,\underline{X}(z) \\
&= \operatorname{diag}\left(\frac{\sum_{m=o}^{M} b_{1m} z^{-m}}{1 + \sum_{n=1}^{N} a_{1n} z^{-n}}, \ldots\right) \underline{X}(z)
\end{aligned} \tag{273}$$

The transfer function for the i-th sensor is thus

$$G_i(z) = \frac{Y_i(z)}{X_i(z)} = \frac{b_o + b_{11}z^{-1} + b_{12}z^{-2} + \ldots + b_{1M}z^{-M}}{1 - a_{11}z^{-1} - a_{12}z^{-2} - \ldots - a_{1N}z^{-N}} \quad (274)$$

The denominator polynomial in (274) determines the stability of the network; it is shown in Ragazzini and Franklin[44] that if the magnitude of the complex roots of the denominator polynomial are less than unity the network characterized by equation (263) is stable. Since the transfer function corresponding to (273) is diagonal and involves no cross-coupling between the individual sensor networks, the question of array stability depends only on the stability of each individual sensor; if all of the sensor networks individually are stable and have finite bounded outputs, their sum will also have finite bound and be stable. However, the problem of stability in (274) is not necessary to consider at the stage of the problem develpment here since the general sensor network in Fig. 10 will be considerably modified in the subsequent development.

The basic Widrow adaptive network is obtained from (274) by setting all of the feedback coefficients a_1 , i=1, ..., N to zero, i.e., the network is a pure feed-forward network such as shown in Fig. 12. Fig. 12 represents a general M-th order feed-forward network and Widrow indicates that the identical delay networks shown will alow the network to receive signals over a band of frequencies if the weights $b_1, \ldots, b_M$ are properly adjusted. In Fig. 13 is shown a single stage of the network with the delay adjusted to $\pi/2$ radians for some input frequency f_1. Since at frequency f_1 the general output would be

$$y(t) = b_o \cos(2\pi f_1 t + \phi_1) + b_1 \sin(2\pi f_1 t + \phi_1) \quad (275)$$

it is clear that such a section could represent perfectly an input signal at frequency f_1 with arbitrary incident phase angle. Returning to Fig. 12, it is clear that M such sections in cascade will allow the exact representation of any input signal which can be written as the finite bandwidth Fourier series

$$y(t) = \sum_{m=-M}^{M} c_m e^{j2\pi m f_1 t} \tag{276}$$

which represents a real bandwidth of Mf_1 Hz. If the delay is not precisely $\pi/2$ radians for the lowest frequency of interest in the input signal the array can still compensate by readjustment of the weights $a_1, \ldots, a_M$ but the magnitudes of the higher order coefficients quite rapidly (in the case of many delays) become very large. More will be said relative to this point later.

Since the matrix B in (273) is diagonal it is convenient in considering the Widrow formulation to define a column weight vector $\underline{w}$ whose components are the diagonal components for the i-th array sensor from B before the application of the Z transform,

$$\underline{w} = \begin{pmatrix} b_o \\ \cdot \\ \cdot \\ b_M \end{pmatrix} \tag{277}$$

Although the whole array of sensors could be handled corporately, it is only necessary to consider the sensors individually since there is no cross-coupling between sensors during the adaptive process to be described.

Defining a vector $\underline{x}_i(t)$ as the array of inputs to the weighting coefficients in Fig. 12, the output $y_i(t)$ can be written as

$$y_i(t) = \underline{w}^*(t)\,\underline{x}_i(t), \tag{279}$$

or in discrete form

$$y_i(k) = \underline{w}^*(k)\underline{x}_i(k) \quad . \tag{280}$$

A very necessary requirement for the use of Widrow's adaptive algorithm is knowledge of the signal $s(t)$, or discretely $s(k)$. If $s(k)$ is known an error signal $\epsilon(k)$, can be defined as the difference between the value $s(k)$ and the present output of the network given by (280),

$$\epsilon(k) = s(k) - \underline{w}(k)^*\underline{x}_i(k). \tag{281}$$

A local gradient search technique, based upon the method of steepest descent, is then used to establish an algorithm for iterating the weight vector $\underline{w}(k)$ to a new value $\underline{w}(k+1)$; the algorithm

$$\underline{w}(k+1) = \underline{w}(k) - 2c_\nu\,\epsilon(k)\,\underline{x}_i(k) \tag{282}$$

can be shown to converge after a large number of iterations to the Wiener-Holf solution. In (282) c_ν is a proportionality constant which is always negative and usually satisfies

$$-1 \le c_\nu \sum_{i=j}^{L} \underline{x}_i{}^*\underline{x}_i < o. \tag{283}$$

The necessity of knowing the signal $s(k)$ which is to be received is the greatest restriction of the algorithm; since the signal is not in general known, except for possibly its second order statistics an a priori assumed artificial signal $s(k)$ must be continually injected into the processor in order for it to adapt. This can be done in two ways. The

artificial signal $s(k)$ can be injected into the processor directly for an adaptive period τ_a during which it cannot be used for receiving, and then the artificial signal is turned off and the performance of the processor "decays" to degraded performance over a period τ_u during which the processor can be used for reception. The mechanism for doing this is shown in Fig. 14. If the power spectral properties of the artificially injected signal $s(k)$ closely approximate those in the incoming waveform $\underline{x}_i(k)$ the τ_u periods will be long relative to the τ_a periods. An alternative method is to provide two processors in parallel; an adaptive processor into which an artificial signal is continually fed to produce the optimum weights $\underline{w}$ to which a second processor is slaved to have the same weights and can thus continually receive external signals. Neither approach appears to be very satisfactory; the on-off approach is clearly useable only part of the time and the continuous, two-processor approach is expensive to implement and is "prejudiced" in favor of the a priori assumed signal $s(k)$.

2. A Visual Display Program

Modifying a program, on which substantial previous programming effort had been contributed by LCDR William Moorehead of the U. S. Navy, an interactive study of the Widrow algorithm was performed. The program employs an IBM 360/67 computer with attached Model (2250-1) display unit; the flow chart for the program is shown in Fig. 15. By means of attention keys located at the display console (see Fig. 16) the program simulation could be interrupted dynamically at any point to provide additional information, change parameters, or produce permanent

Calcomp plots of any of the console displayed results.

The basic information required for input to the program is the assumed direction and spectrum (center frequency and bandwidth) of the desired signal. The desired input signal power is automatically adjusted by the program to be unity with uniform power density being spread among the various sideband components over the desired bandwidth. The actual signal is then synthesized by use of the inverse discrete FFT algorithm.

The noise field in the experiments that were performed was assumed to be incident normally to the array (see Fig. 17; as such the effective aperture of the noise signal is greater than that of incident signals for all incident signal directions not also normal to the array. Widrow employed sinusoidal "noise" for his test examples; the present investigation used random noise. The additive random Gaussian noise was added to the array from a random number generator program which produced random numbers of zero mean and known standard deviation.

The time delay taps on the delay line were variable from a minimum of every sampling interval to any larger integral multiple of the basic sampling interval.

At any given time in the adaption procedure up to four plots of directivity pattern vs. frequency and four plots of frequency response vs. angle could be obtained. This allows the receiving patterns at frequencies different from the desired frequency and frequency response diagrams for angles different from the desired look angle to be obtained

simultaneously. In addition the program provides the output spectrum of the array filter and the array filter weights $\underline{w(k)}$.

In Appendix F are shown the results of running the program for the physical situation illustrated in Fig. 17.

In general the results are as follows. The filter provides excellent suppression for all frequencies outside of the band $400 < f < 600$ cps. in the desired look direction of $\theta = 45^\circ$. The response of the filter to other frequencies and angles of incidence is much reduced with the exception of frequencies above 1000 H_z where the pattern exhibits the familiar structure of linear arrays with element spacings greater than $\lambda/2$. i.e., the many lobe problem. This problem can be easily overcome, however, by either closer array element spacing or by ignoring the pattern and placing low pass filters of pass-band, $o \leq$ pass-band $\leq 2f_1$ where f_1 is the desired signal frequency of interest.

In addition it is clear that the adaption time for the filter to reach steady state is quite long for general white noise inputs of the type used here; much longer than for the narrow band simusoidal noise used in Widrow's paper.[40]

3. Other Approaches: Future Research

The results reported in this paper relative to the Widrow filter simulation program are minimal; rather it is thought more appropriate to make a few generalizations and suggestions for future research based upon results already observed.

The Widrow filter must have knowledge of the signal to be detected (at least the nominal frequency and bandwidth); since a priori the signal

to be detected is of course not known an assumed signal s(k) must be applied to adapt the filter. From a practical point of view this must mean a continual process of adaption using an ensemble of maximum likelihood input signals s(k) as the adaptive signal inputs; the speed of adaption is thus of fundamental importance as all angles, bandwidth frequencies, and nominal center frequency ranges of the ensemble must be scanned during the acquisition phase of operation. The single-mode two-processor adaption algorithm is thus almost a necessity.

Secondly, the real time speed of adaption is strongly a function of the amount of adaption computations the processor is required to do. In this regard the alternative procedure of assuming every input is narrow-band to first order during the acquisition phase (and thus requires only the computations for b_o, b_1 for any element) appears to have special merit, especially for the detection of amplitude modulated signals. During the acquisition phase $b_2, \ldots, b_M$ would be maintained at zero. After acquisition of a signal the "fine structure" of the broad-band spectrum can be provided by adding other spectral components to the adaption signal and adapting the weights $b_o, b_1, b_2, \ldots, b_M$ as before. The signal output during the preliminary adaption phase would be quite distorted; this might be a tolerable price to pay for much faster speed of adaption.

Another possibility for increased speed of adaption as well as providing more stable weights would appear to be the use of non-uniform time delays. There is nothing in the recursive algorithm (276) which requires that the delays all be equal; the algorithm requires only that the

weight vector, error scalar, and particular input signal be known at any sequential time step. Consider for example the amplitude modulation case in which the desired signal is as shown in Fig. 18. This signal s(t) is wide-band in the sense that the (band-width) / (center-frequency) $\sim .5$ but is narrow-band in the sense of equation (276). If all of the delays are of size $1/4f_1$ only the fundamental is very effective in simulating the input signal; the amounts of the higher order harmonics in (276) provide very small envelope corrections to the waveform synthesized by b_o and b_1. This suggests the network shown in Fig. 19; the adaption procedure consists of first doing a first-order narrow-band adaption as previously described to obtain b_o and b_1 for the first delay of $1/4f_1$ with b_2, ..., b_M fixed at zero and the desired adaption signal being a monochromatic signal of frequency f_1. When the gradient of the error function $\epsilon(k)$ decreases to a pre-assigned threshold value, the additional signal components are added to the desired adaption signal s(t) and the weights b_o, b_1, b_2, ..., b_M are all allowed to adapt to steady state value. If the desired signal s(t) is of the form shown in Fig. 18 only a very few additional weights b_2, ..., b_M should be required and the adaption time should be very rapid.

4. The Use of Feedback Networks: The Kalman Filter

The work of Mantey[41] indicates that it may be possible to produce a better adaptive synthesis procedure using feedback networks. However, he shows that the sum-squared quadratic error function

$$\sum_{k=o}^{\infty} [\epsilon_i(k)]^2 = \sum_{k=o}^{\infty} [s_i(k)-y_i(k)]^2 \qquad (284)$$

possesses a unique minimum with respect to adaption only if the feedback coefficients $a_1, \ldots, a_N$ are all fixed. He then shows that modified adaption procedure in which the desired signal is used directly as the adaption signal applied to the feedback coefficients (Fig. 20), leads to a quadratic performance criterion which has a unique minimum. This new adaption procedure, in the case that x(k) is white, has also been shown to be a stable network.[41,22]

Thus it would appear that perhaps substantial improvements in adaption performance might be realized by use of time variable feedback networks employing one or more feedback loops. The Kalman filter[46,47] is such a filter; it is a least-square-estimation filter which provides optimal estimates of the state vector s(k) recursively. The problem may be formulated here as that of obtaining the best estimate $\hat{\underline{s}}(k/k)$, of the state vector $\underline{s}(k)$ at time state k given k previous observations x(k) and the previous optimal estimate $\hat{\underline{s}}(k/k-1)$ of the state vector. It is assumed that $\underline{s}(k)$ satisfies the dynamic system relations

$$\underline{s}(k) = \Phi(k,k-1)\,\underline{s}(k-1) + \Delta(k,k-1)\underline{n}(k) \tag{285}$$

$$\underline{x}(k) = H(k)\underline{s}(k) + \underline{v}(k) \tag{286}$$

where:

$\underline{s}(k)$ is the state vector (unknown) at time state k

$\Phi(k,k-1)$ is the state transition matrix (known)

$\Delta(k,k-1)$ is a known transition matrix

H(k) is a known transformation matrix

$\underline{x}(k)$ is the observation vector

$\underline{n}(k)$, $\underline{v}(k)$ are sequences of independent Gaussian random vectors with zero means and the known covariance matrices

$$E[\underline{n}(k)\,\underline{n}^*(\ell)] = Q\,\delta_{k,\ell} \quad E[\underline{v}(k)\,\underline{v}^*(\ell)] = R\,\delta_{k,\ell}$$

$$E[\underline{n}(k)\,\underline{v}^*(\ell)] = \underline{o}\ . \tag{287}$$

They represent assumed additive excitation and measurement noise signals respectively.

Following Lee,[47] the optimal estimate $\hat{\underline{s}}(k/k)$ is assumed to be the sum of the extrapolated estimate, based on past estimates and a weighting "gain" G(k), times the residual difference between the new measurement and the extrapolated previous measurement,

$$\hat{\underline{s}}(k/k) = \Phi\hat{\underline{s}}(k/k-1) + G(k)\,[x(k) - H\hat{\underline{s}}(k/k-1)]\ . \tag{288}$$

This can be re-written in the form

$$\hat{\underline{s}}(k/k) = [\Phi - GH\,\Phi]\,\hat{\underline{s}}(k-1/k-1) + G\,\underline{x}(k)\ . \tag{289}$$

This suggests in the case of two equal time delays the state transition flow diagram shown in Fig. 21. The linear equations equivalent to (289) are, from Fig. 21,

$$\hat{s}_1(k/k) = [\phi_{11} - g_{11}\phi_{11}]\hat{s}_1(k-1/k-1) + [\phi_{12} - g_{11}\phi_{12}]\hat{s}_2(k-1/k-1) + g_{11}x(k) \tag{290}$$

$$\hat{s}_2(k/k) = [\phi_{21} - g_{21}\phi_{11}]\hat{s}_1(k-1/k-1) + [\phi_{22} - g_{21}\phi_{12}]\hat{s}_2(k-1/k-1) + g_{21}x(k) \tag{291}$$

$$\hat{s}_1(k-1/k-1) = z^{-1}\hat{s}_1(k/k) \tag{292}$$

$$\hat{s}_2(k-1/k-1) = z^{-1}\hat{s}_2(k/k) \tag{293}$$

where

$$G = \begin{pmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{pmatrix}, \quad H = \begin{pmatrix} 1 & o \\ o & o \end{pmatrix}, \quad \Phi = \begin{pmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{pmatrix}. \tag{294}$$

Since the desired output is $\hat{s}_1(k/k)$, the system (290,293) can be solved for the transfer function $\hat{s}_1(k/k) / x(k)$,

$$\frac{\hat{s}_1(k/k)}{x(k)} = \frac{g_{11}[1-(\phi_{22}-g_{21}\phi_{12})z^{-1}] + g_{21}[\phi_{12}]z^{-1}}{\Psi_1}$$

$$= \frac{g_{11}[\phi_{22}-g_{21}\phi_{12}-\phi_{12}+g_{11}\phi_{12}]z^{-1}}{\Psi_2} \qquad (295)$$

where

$$\Psi_1 = (1-(\phi_{11}-g_{11}\phi_{11})z^{-1})(1-(\phi_{22}-g_{21}\phi_{12})z^{-1})-(\phi_{12}-g_{11}\phi_{12}) \cdot (\phi_{21}-g_{21}\phi_{11})z^{-2}$$

$$\Psi_2 = 1-[\phi_{11}-g_{11}\phi_{11}+\phi_{22}-g_{21}\phi_{12}]z^{-1} + [(\phi_{11}-g_{11}\phi_{11})(\phi_{22}-g_{21}\phi_{12}) - (\phi_{12}-g_{11}\phi_{12})(\phi_{21}-g_{21}\phi_{11})]z^{-2} .$$

This, by comparison with (274) is equivalent to the sample data system shown in Fig. 22. For any desired signal s(k) it is possible to thus obtain the optimal least-squares estimate of $\hat{s}_1(k/k)$ at the output of the network provided

$$b_o(k) = g_{11}(k) \qquad (298)$$

$$b_1(k) = [\phi_{22}(k)-g_{21}(k)\phi_{12}(k)-\phi_{12}(k)+g_{21}(k)\phi_{12}(k)] \qquad (299)$$

$$a_1(k) = [\phi_{11}(k)-g_{11}(k)\phi_{11}(k)+\phi_{22}(k)-g_{21}(k)\phi_{12}(k)] \qquad (300)$$

$$a_2(k) = [(\phi_{11}(k)-g_{11}(k)\phi_{11}(k))(\phi_{22}(k)-g_{21}(k)\phi_{12}(k)) - (\phi_{12}(k)-g_{11}(k)\phi_{12}(k))(\phi_{21}(k)-g_{21}(k)\phi_{11}(k))] . \qquad (301)$$

If the signal process s(k) is stationary, the matrix Φ will be

time invariant and the gain matrix G(k) will quite soon reach steady states; this leads to the exciting possibility that the steady state pre-calculated values for a_1, a_2, b_o, b_1 may be used directly in (295). Although this would not give optimal estimates during the usual transient phase of the adaption procedure, previous experience with the technique of using fixed gains indicates that it begins providing optimal estimates after a very short time.

VII. CONCLUSIONS

This research has concerned investigations into the inter-related topics of large-domain integration techniques for solution to the scalar multi- dimensional variable-parameter acoustic wave equation and adaptive antenna-array processors and algorithms.

After formulation of the acoustic propagation medium equations to be integrated in properly posed form, their solution by means of various finite-difference integration technqiues was investigated. Among the results that are considered new or significant are the following.

A method for "tearing"[45] or partitioning very large variable parameter domains in cases where steady state propagation characteristic are desired is given. Such positioning is essential if the integration of the wave equation for large domains is to be carried out on reasonable-sized machines or in background mode in a multi-tasking environment. Computer calculations verify that consistent results are obtainable using the method, but its practical use depends upon the efficiency of the integration scheme chosen and "asking the right questions." Since, inherently, the method is capable of giving the complete steady-state pressure velocity distribution profiles at all points of space it is not surprising that quite large computing time requirements result. However, if the integration is confined by program control to only those regions of the spacial domain wherein the state variables are above an assigned threshold, great reductions in computing time are possible. Indeed there appears no a priori reason why ray-path analysis cannot be used for regions wherein the properties are known to be quite regular with a switch to direct

wave-equation integration when approaching regions of unpredictable behavior such as the incidence of convergence zones, highly variable parameter domains, defraction regions, or where three-dimensional behavior cannot be ignored.

A new method for integrating the wave-equation in three dimensions is developed which uses the Fast Fourier Transform digital algorithm. The method is fast, accurate, and easily implemented for general rectangular domains but as with all Fourier techniques is restricted to constant parameter regions. The method is semi-implicit and appears to possess considerable stability even for fairly large integration step sizes.

The Douglas and Gunn tri-diagonal integration scheme was implemented for the wave equation in three dimensions. The scheme is faster computationally than any of the other algorithms tried in the case of variable coefficient problems. However, it requires a great deal of programming effort, large amounts of work storage, and the conditions of stability are not precisely known.

As an aid in understanding and applying the general discrete Fourier Transform, a new method for finding analytic closed form expressions for the coefficients is given. The method is basically an extension to discrete domains of the impulse function method of continuous distribution theory. Although the method does not guarantee that the investigators efforts will be small, it appears that coefficient expressions can be obtained quite easily for functions having relatively few discontinuities in analytic-envelope.

Relative to processors which seem especially appropriate for use in variable parameter environments the topic of adaptive array processors and algorithms was considered. First, the classical optimum statistical estimation procedure as it applies to array processors was reviewed. In the classical analog approach it is shown that discrete spectrum algorithms of the FFT type can be substituted provided in the analogous continuous spectrum variables proper precautions are observed relative to estimation and/or detection interval T and sampling interval Δt. This generally will require that N, the number of sampling points for dection interval T, be quite large; this is no particular restriction and in fact the FFT algorithm approaches maximum efficiency for large N.

The second point of view is that leading to automatic adaptive synthesis procedures in which quadratic-error functional minimizations are carried out. The Widrow scheme is considered as a special case of a general feed-forward, feedback network of the type considered extensively by Mantey. Results of a visual display simulation program are reported and the use of other possible adaptive algorithms such as variable delays and the Kalman filter are discussed.

APPENDIX A

THE DOUGLAS AND GUNN EQUATIONS

The Douglas and Gunn equations iteratively solve the wave equation in three dimensions by computing a series of three approximations to the state components at the time $(r+1)\,\Delta t$ in terms of state components at the time $r\,\Delta t$. The particular equations to be solved for any one approximation are tri-diagonal in terms of the implicit variables; the solution to the implicit portion of the previous approximation acts as an explicit input to the next approximation. Thus the implicit system of equations to be solved at any stage are tri-diagonal and hence, in the case of the three dimensional wave equation the work of solving the system is only slightly greater than three times the work required to solve the equivalent one-dimensional wave equation.

Denoting by $\underline{u}'(\ell, m, n, r)$ the known state of the system at position $\ell\,\Delta x$, $m\,\Delta y$, $n\,\Delta z$, $r\,\Delta t$, by $\underline{u}^*(\ell, m, n, r+1)$ the first approximation to the solution of the wave equation at state $(r+1)\,\Delta t$, by $\underline{u}^{**}(\ell, m, n, r+1)$ the second approximation, etc., the equations to be solved are shown below. For purposes of brevity only the indices that vary in any given term are explicitly written down.

$$p^*(r+1)-p = -\alpha\,[v_x^*(\ell+1,r+1)-v_x^*(\ell-1,r+1) + v_x(\ell+1) - v_x(\ell-1) + 2\,(v_y(m+1) - v_y(m-1) + v_z(m+1) - v_z(n-1))\,] \quad \text{(A1)}$$

$$v_x^*(r+1)-v_x = -\alpha\,[p^*(\ell+1,r+1)-p^*(\ell-1,r+1) + p(\ell+1) - p(\ell-1)\,] \quad \text{(A2)}$$

$$v_y^*(r+1) - v_y = -2\alpha\,[p(m+1) - p(m-1)\,] \quad \text{(A3)}$$

$$v_z^*(r+1) - v_z = -2\alpha\,[p(n+1) - p(n-1)\,] \quad \text{(A4)}$$

End of first approximation equations

$$p^{**}(r+1) - p = -\alpha\,[v_x^*(\ell+1,r+1)-v_x^*(\ell-1,r+1) + v_x(\ell+1) - v_x(\ell-1) + v_y^{**}(m+1,r+1) - v_y^{**}(m-1,r+1) + v_y(m+1)-v_y(m-1) + 2(v_z(n+1) - v_z(n-1))\,] \quad \text{(A5)}$$

$$v_x^{**}(r+1) - v_x = -\alpha\,[p^*(\ell+1,r+1) - p^*(\ell-1,r+1) + p(\ell+1) - p(\ell-1)\,] \quad \text{(A6)}$$

$$v_y^{**}(r+1) - v_x = -\alpha\,[p^{**}(m+1,r+1)-p^{**}(m-1,r+1)+p(\ell+1)-p(\ell-1)\,] \quad \text{(A7)}$$

$$v_z^{**}(r+1) - v_z = -2\alpha\,[p(n+1)- p(n-1)\,] \quad \text{(A8)}$$

End of second approximation equations

$$p(r+1)-p = -\alpha\,[v_x^*(\ell+1,r+1)-v_x^*(\ell-1,r+1) + v_x(\ell+1)-v_x(\ell-1) + v_y^{**}(m+1,r+1) - v_y^{**}(m-1,r+1)+v_y(m+1)-v_y(m-1) + v_z(m+1,r+1)-v_z(n-1,r+1)+v_z(n+1)-v_z(n-1)\,] \quad \text{(A9)}$$

$$v_x(r+1)-v_x = -\alpha\,[p^*(\ell+1,r+1)-p^*(\ell-1,r+1)+p(\ell+1)-p(\ell-1)\,] \quad \text{(A10)}$$

$$v_y(r+1)-v_y = -\alpha\,[p^{**}(m+1,r+1)-p^{**}(m-1,r+1)+p(m+1)-p(m-1)\,] \quad \text{(A11)}$$

$$v_z(r+1)-v_z = -\alpha\,[p(n+1,r+1)-p(n-1,r+1)+p(n+1)-p(n-1)\,] \quad \text{(A12)}$$

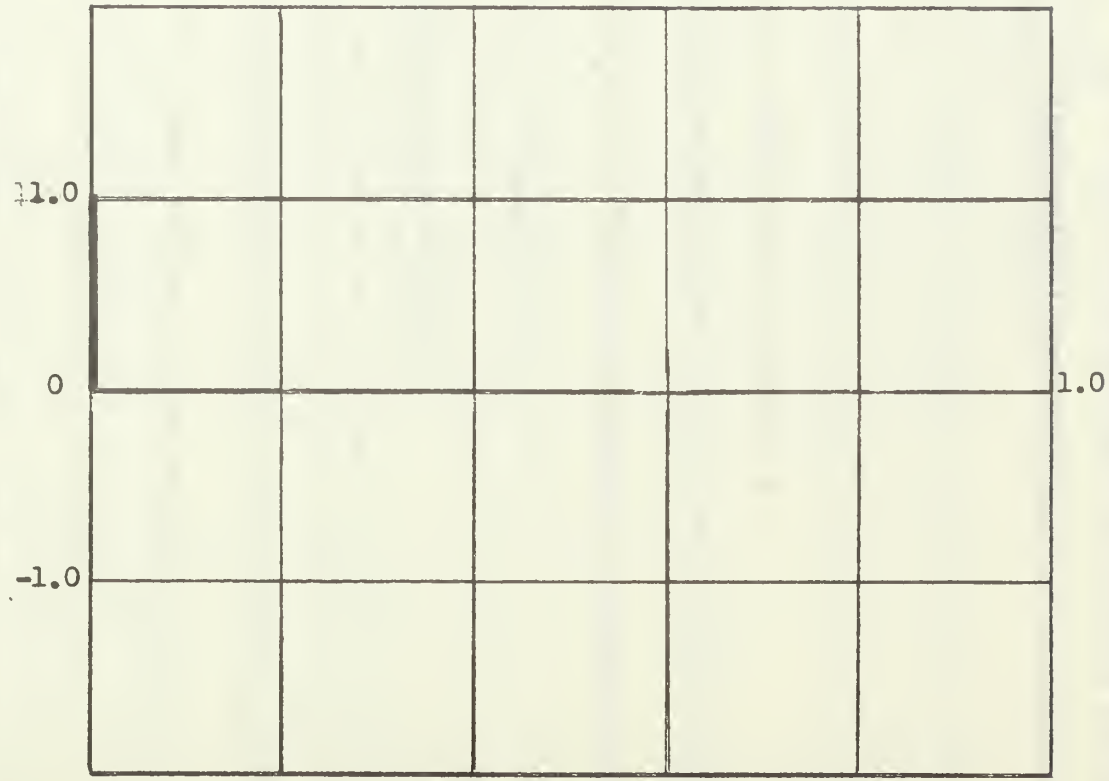
where

$$\alpha = -\frac{c\,\Delta t}{4\,\Delta x} \quad . \quad \text{(A13)}$$

APPENDIX B

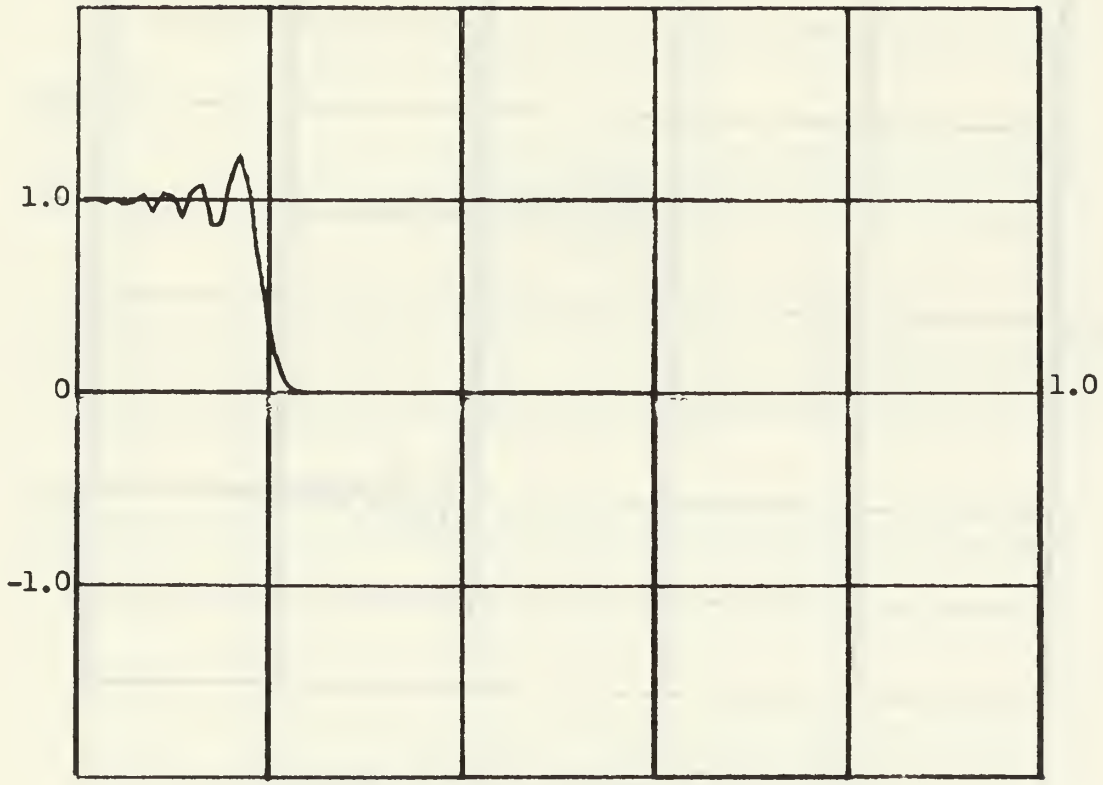
THE SIGNAL INPUT TO A MATCHED LINE

This appendix shows the response of a matched output one-dimensional domain to a step input function applied at time t=o sec. The domain is of unit length and is subdivided into 100 segments.



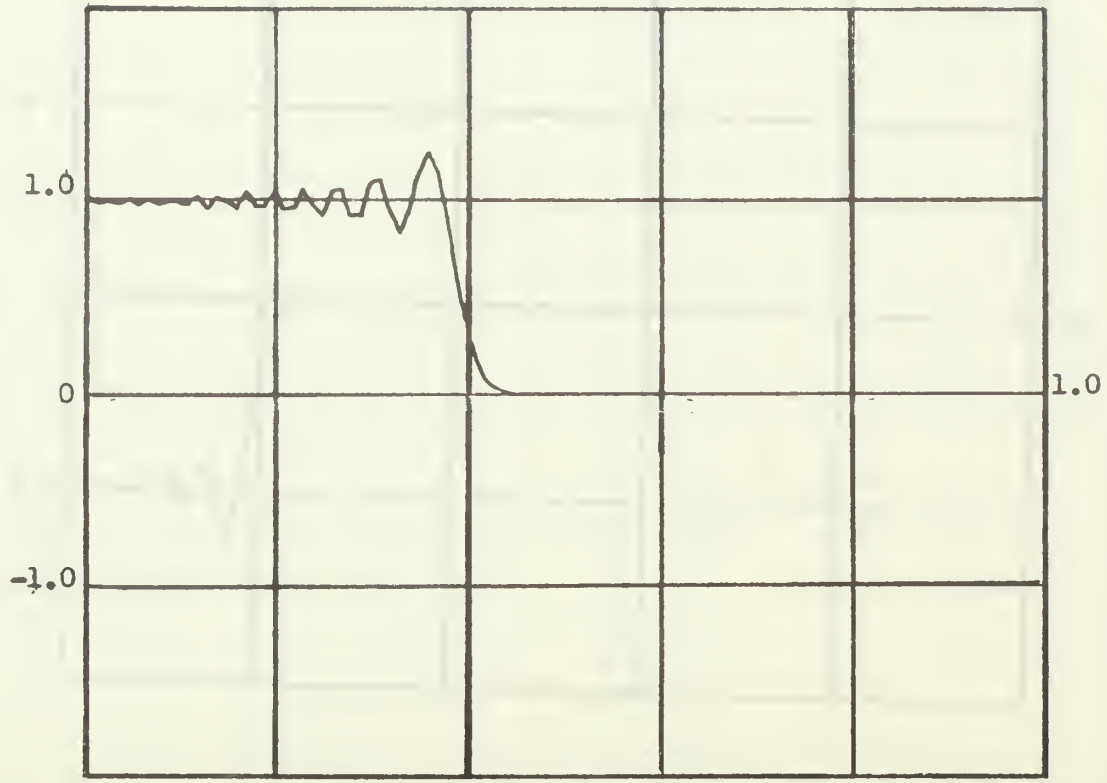
Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t = 0 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



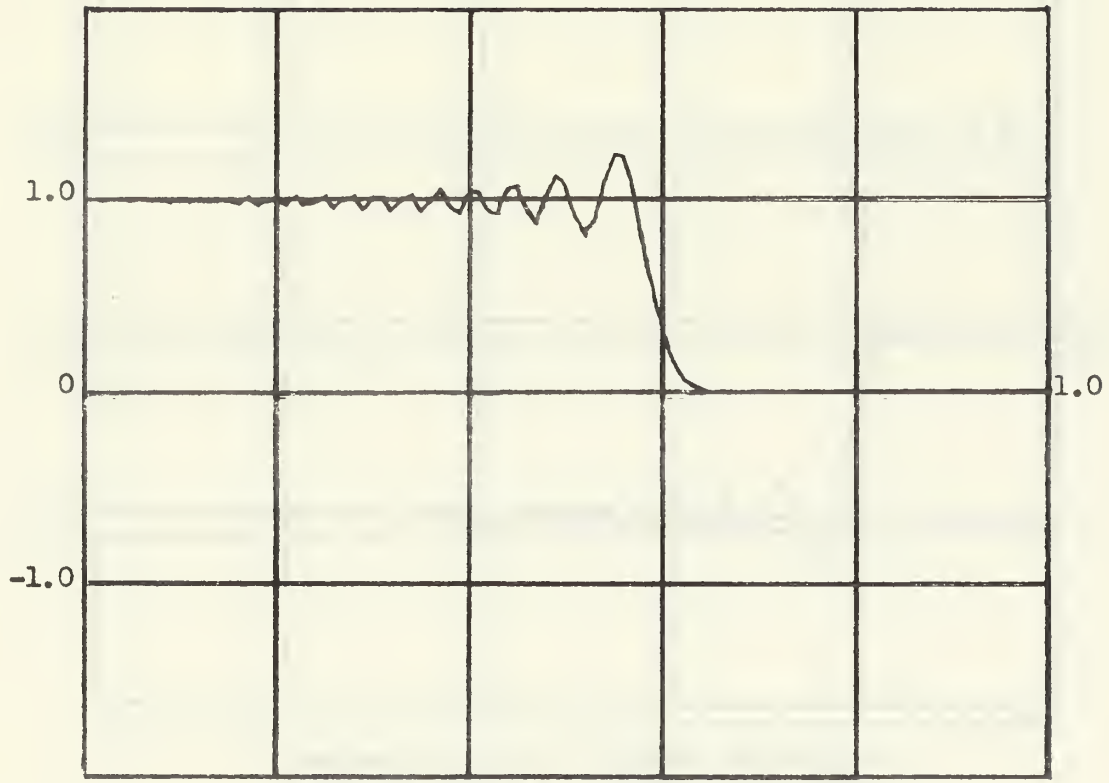
Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t = .20 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



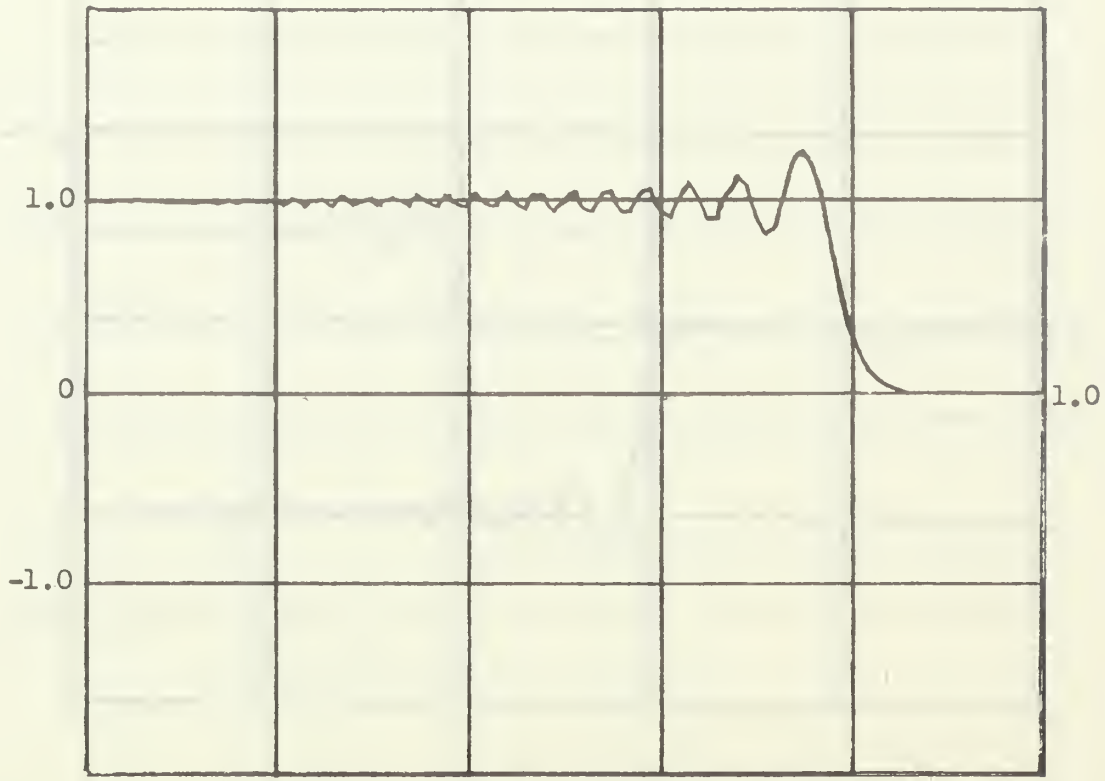
Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t = .40 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01

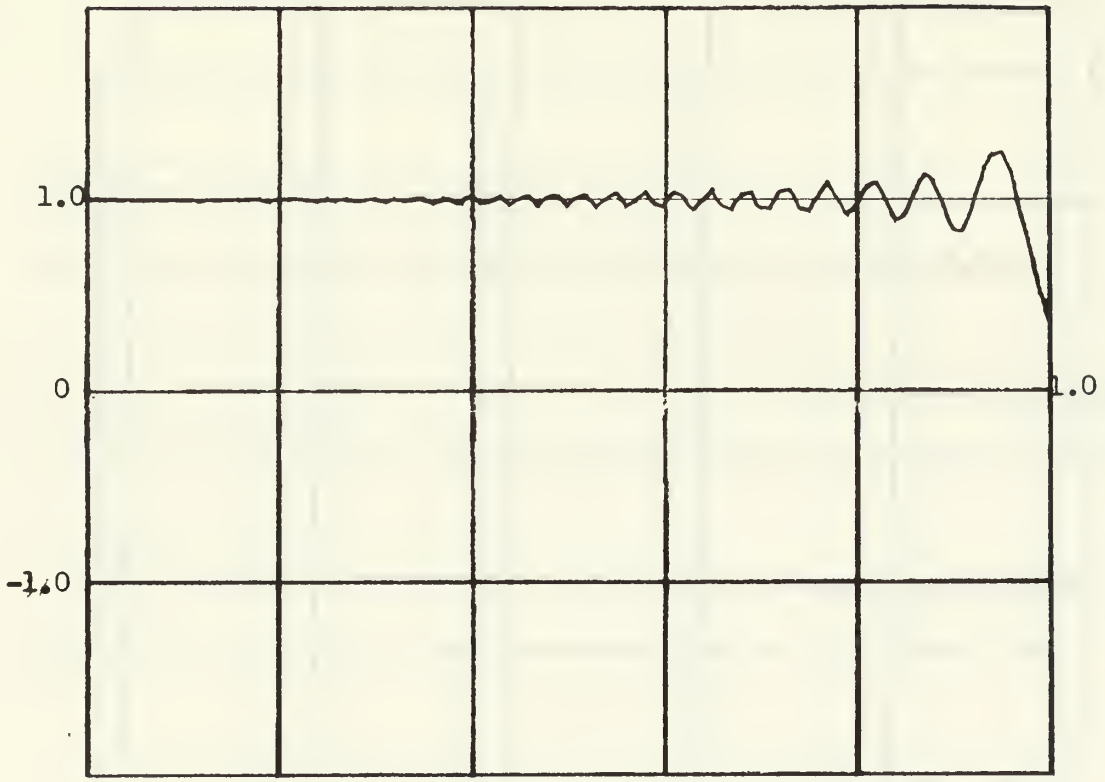


Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t =.60 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01

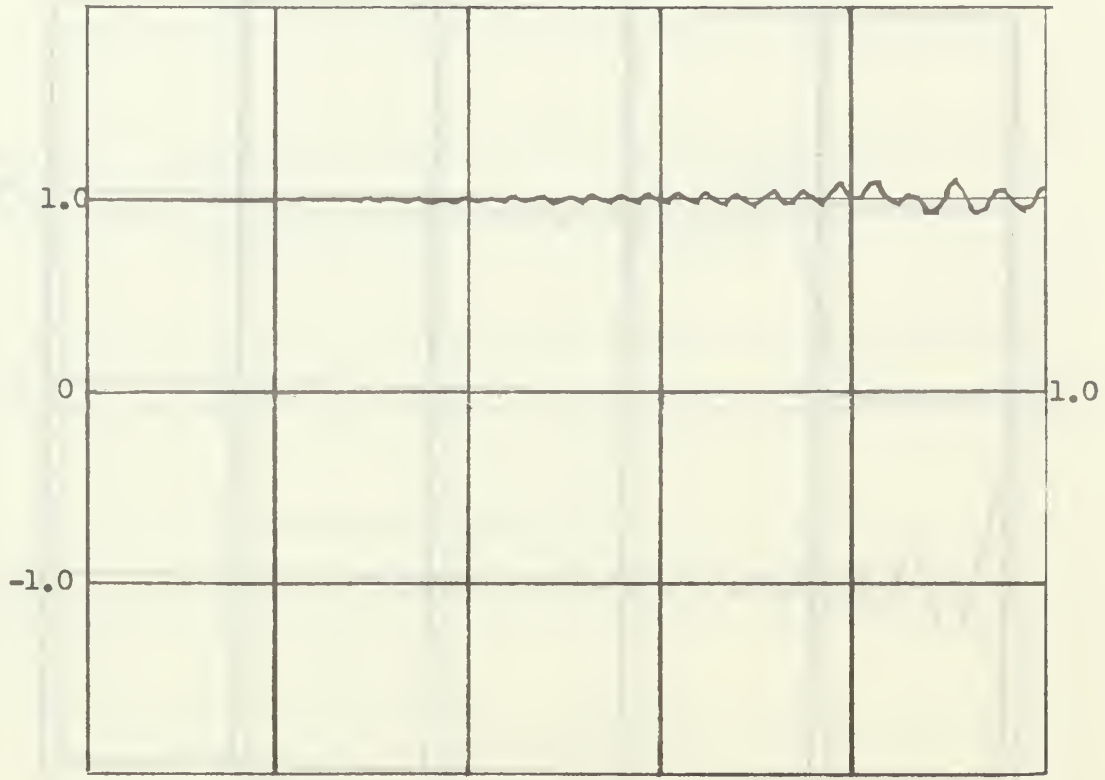


Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch
Spacial pressure distribution at t = .80 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01

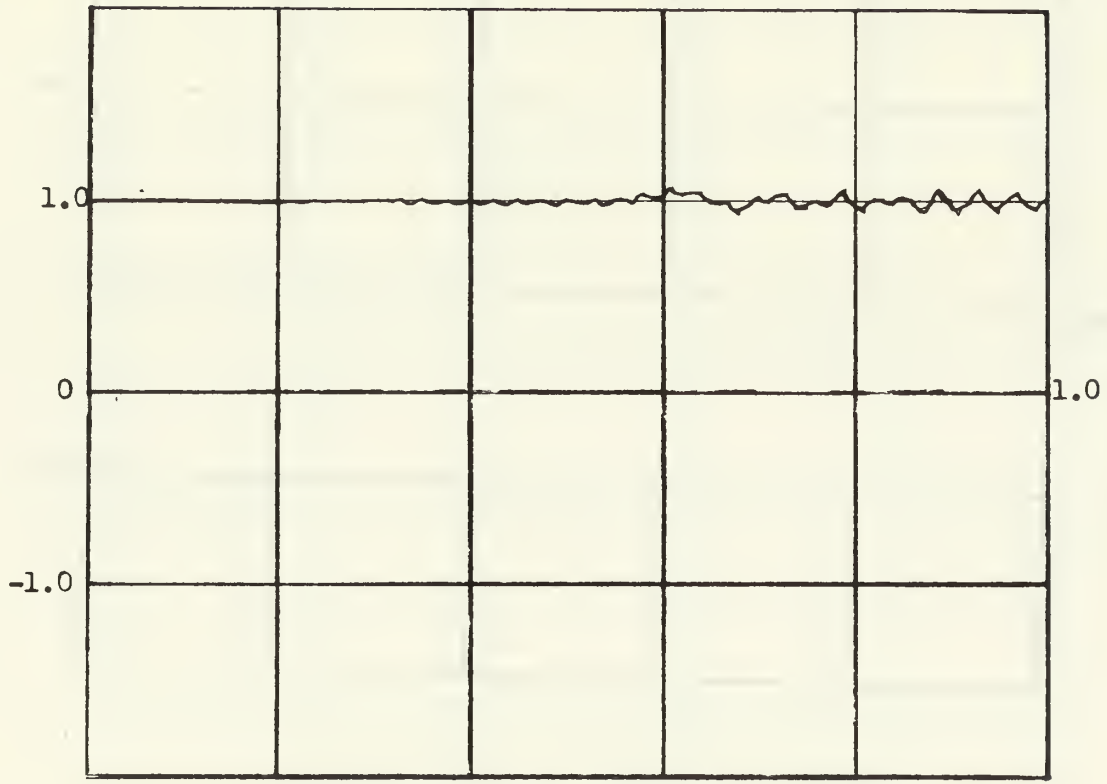


Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

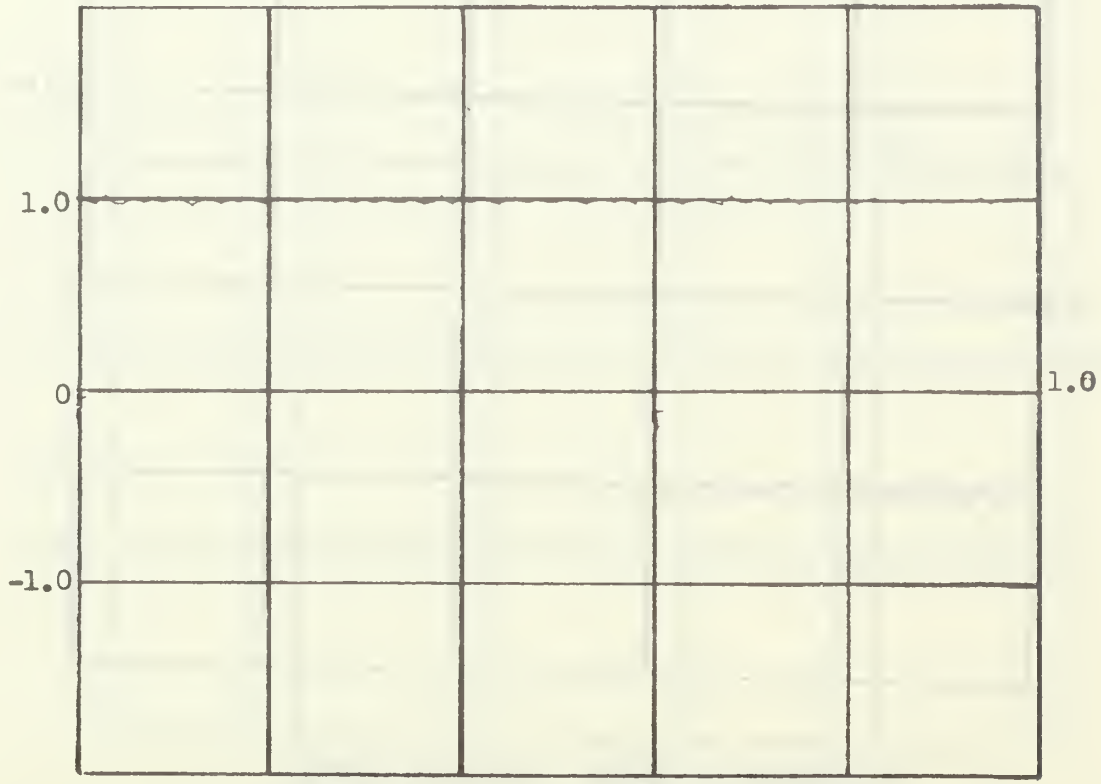
Spacial pressure distribution at t = 1.0 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch
Spacial pressure distribution at t = 1.20 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E-01 units/inch
Spacial pressure distribution at t = 1.40 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



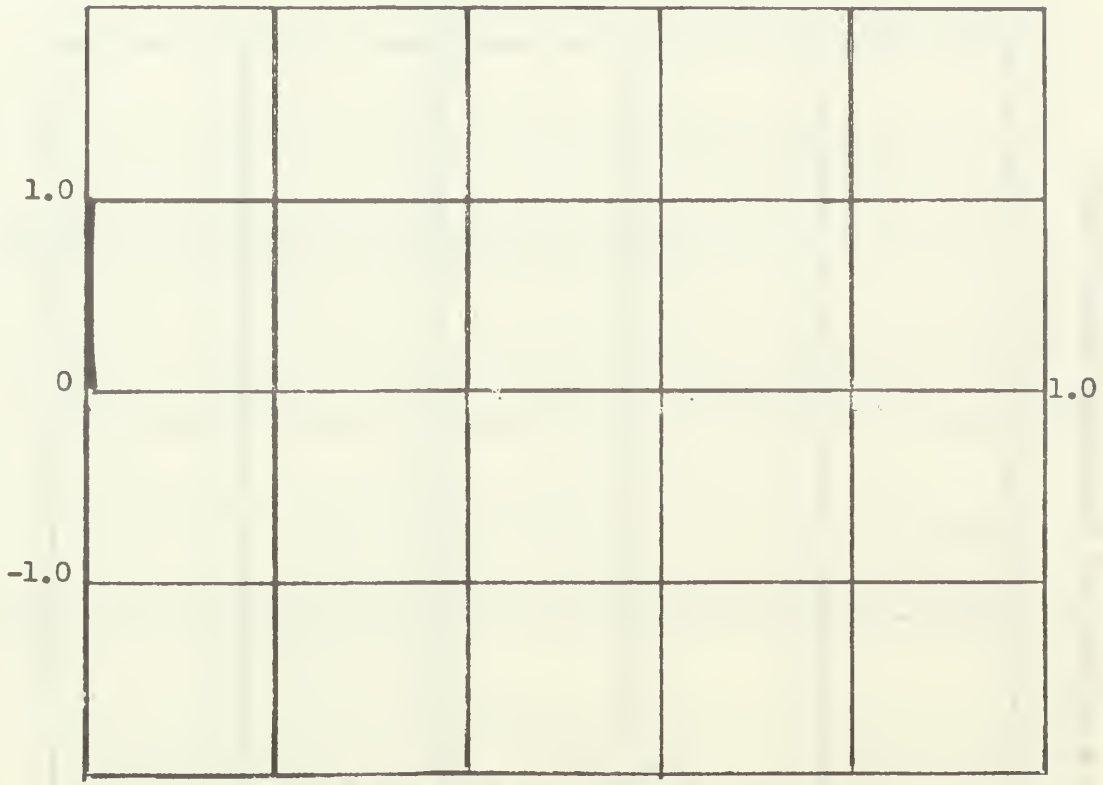
Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t = 6.0 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01

APPENDIX C

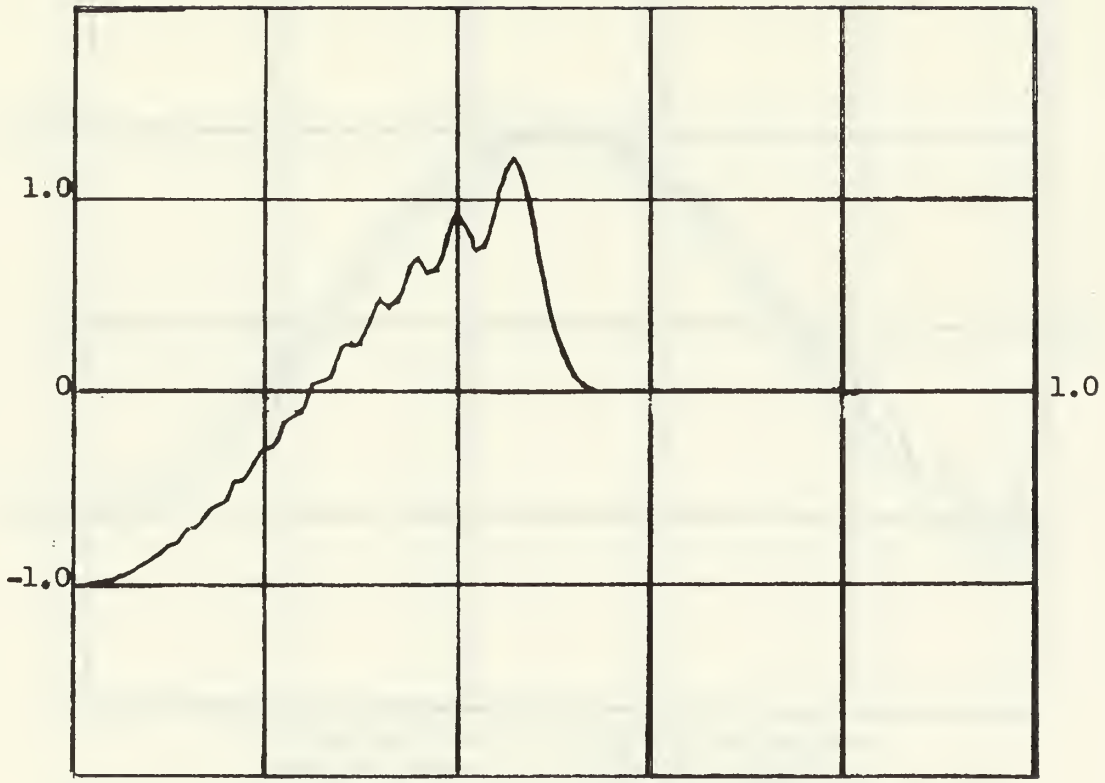
THE COSINE INPUT TO A MATCHED LINE

This appendix shows the response to the same domain as described in Appendix B but with a 1 Hz cosine signal input.



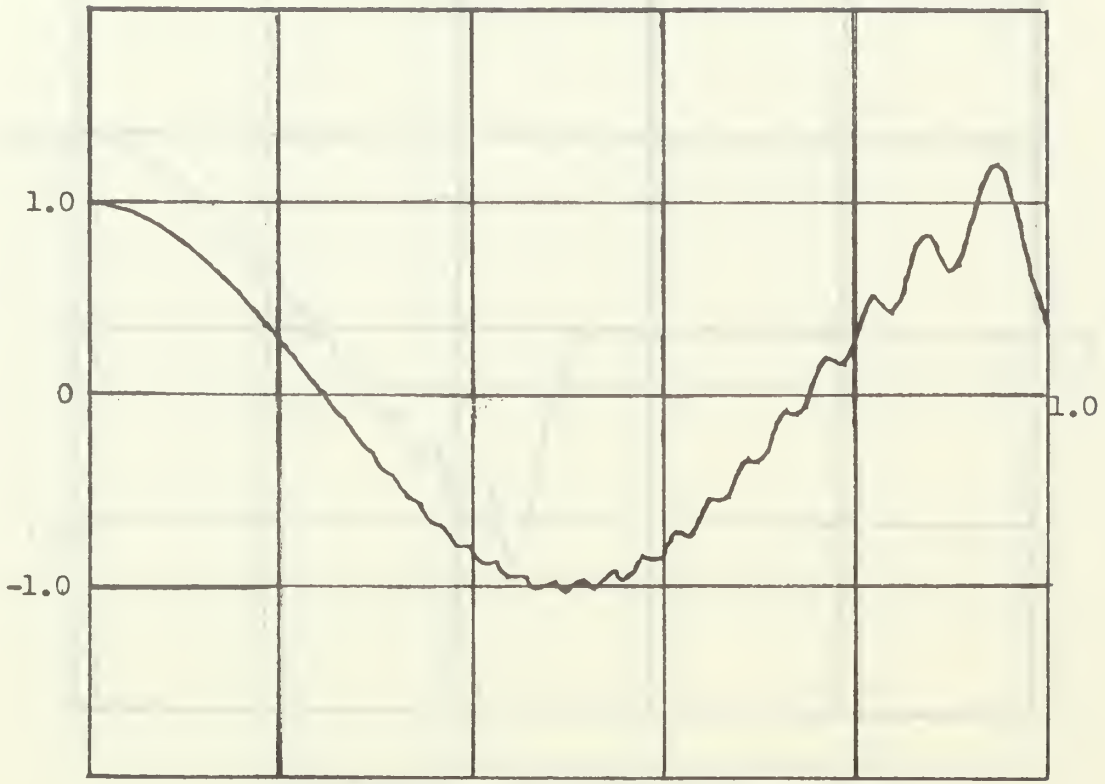
Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t = 0.00 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



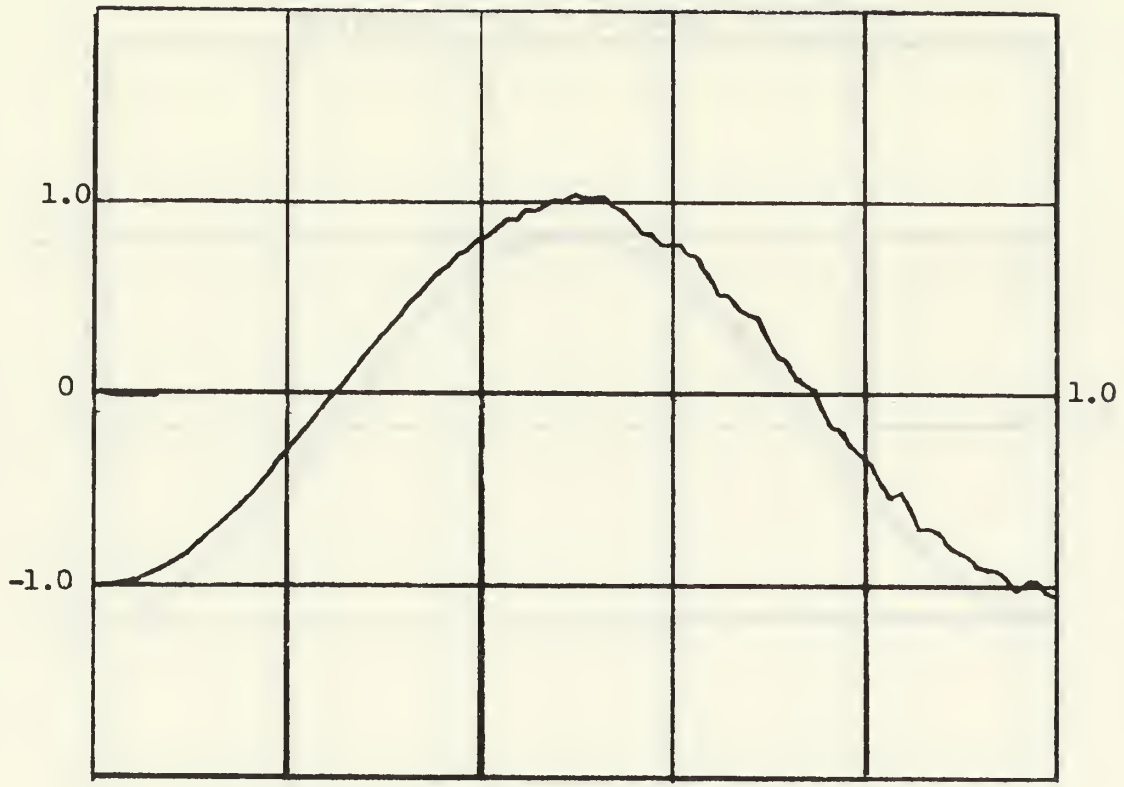
Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t = .50 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01

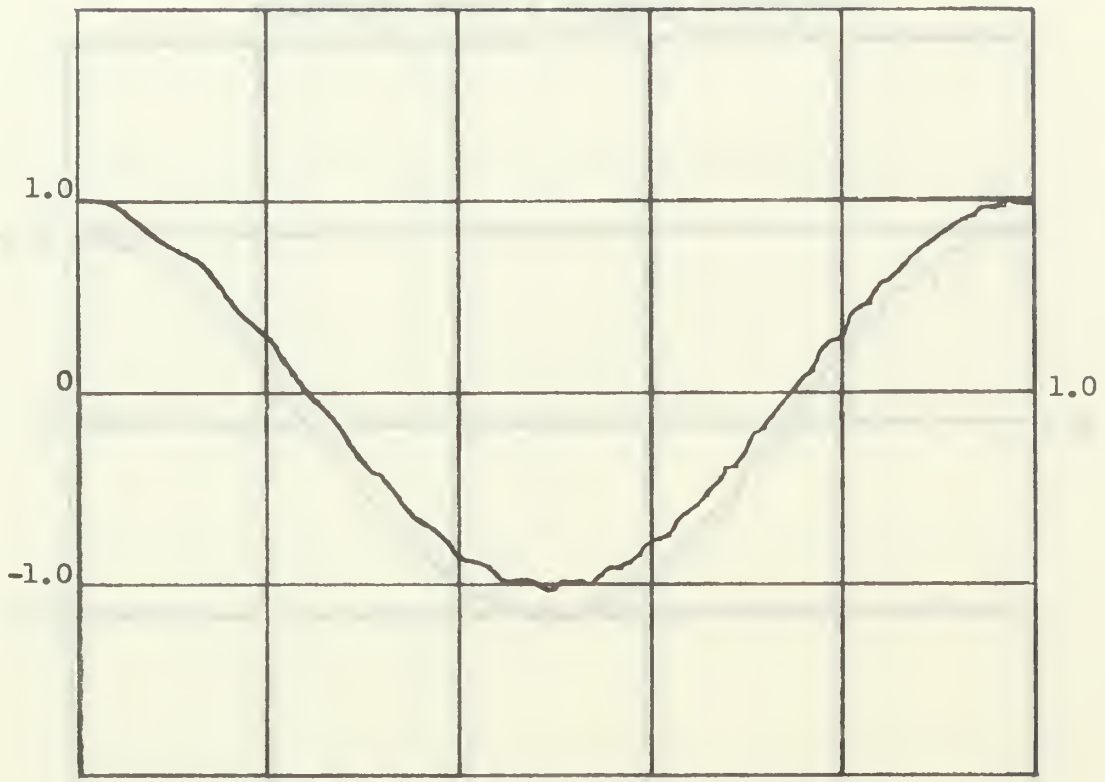


Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t = 1.00 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01

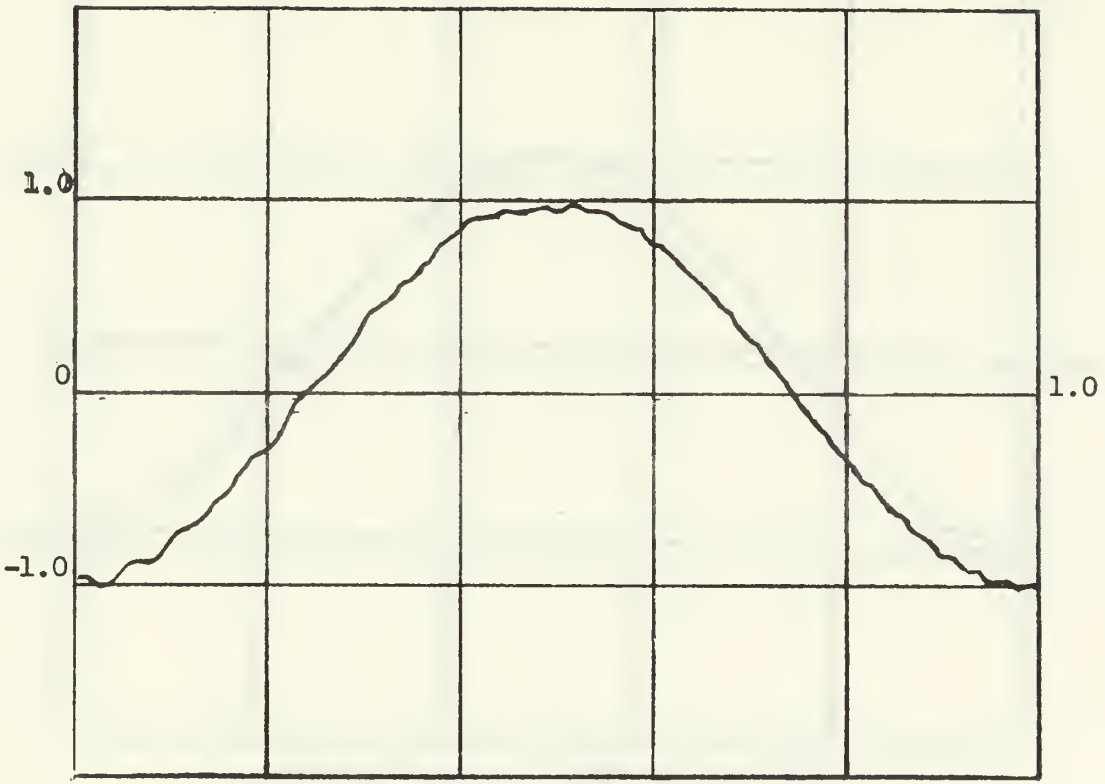


Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch
Spacial pressure distribution at t = 1.50 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

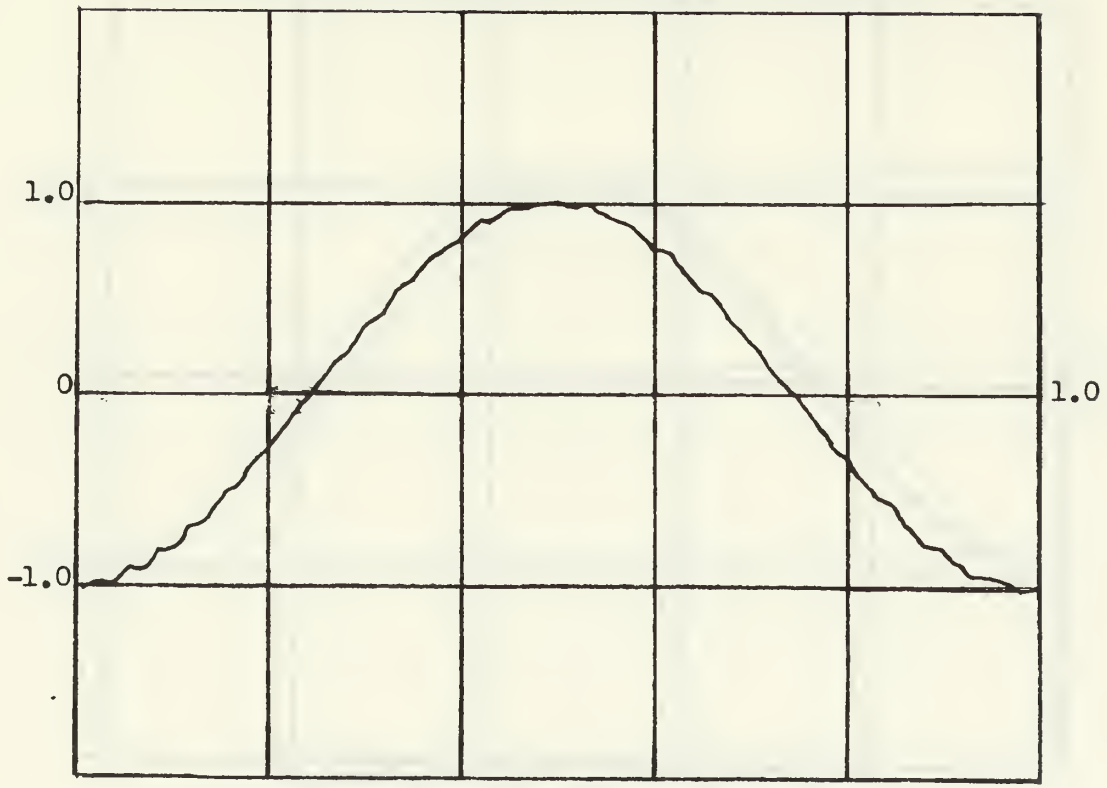
Spacial pressure distribution at t = 2.00 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

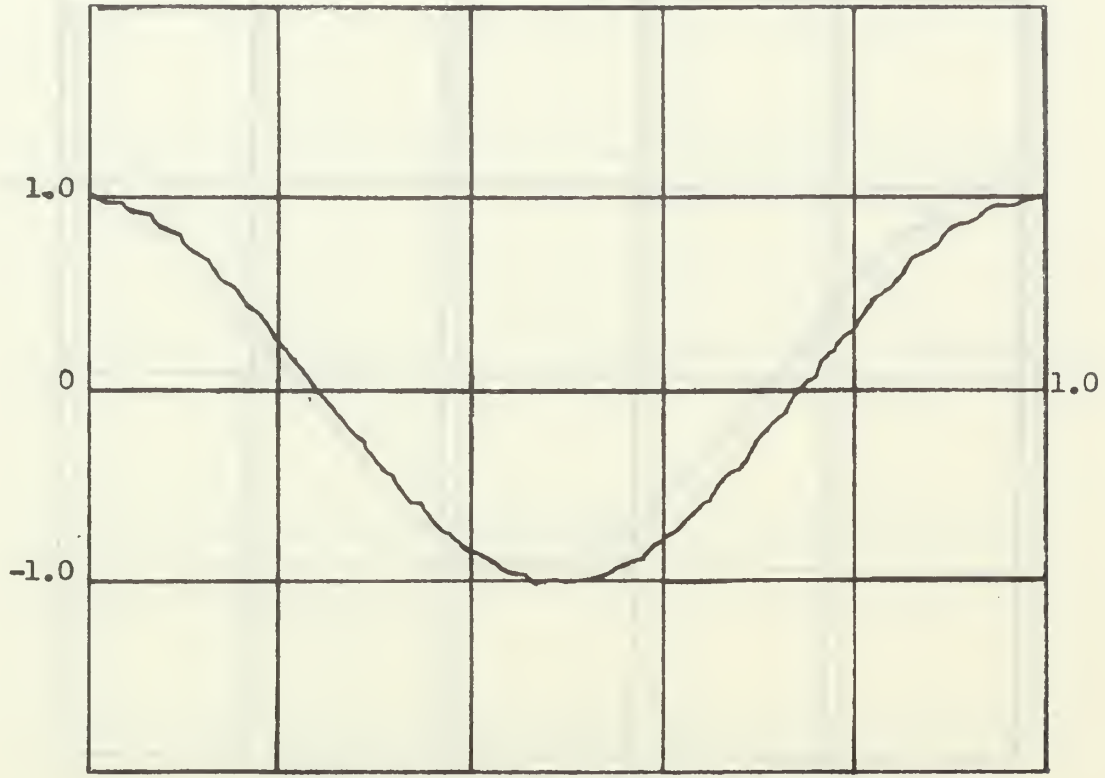
Spacial pressure distribution at t = 2.50 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01

Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch
Spacial pressure distribution at t = 3.00 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



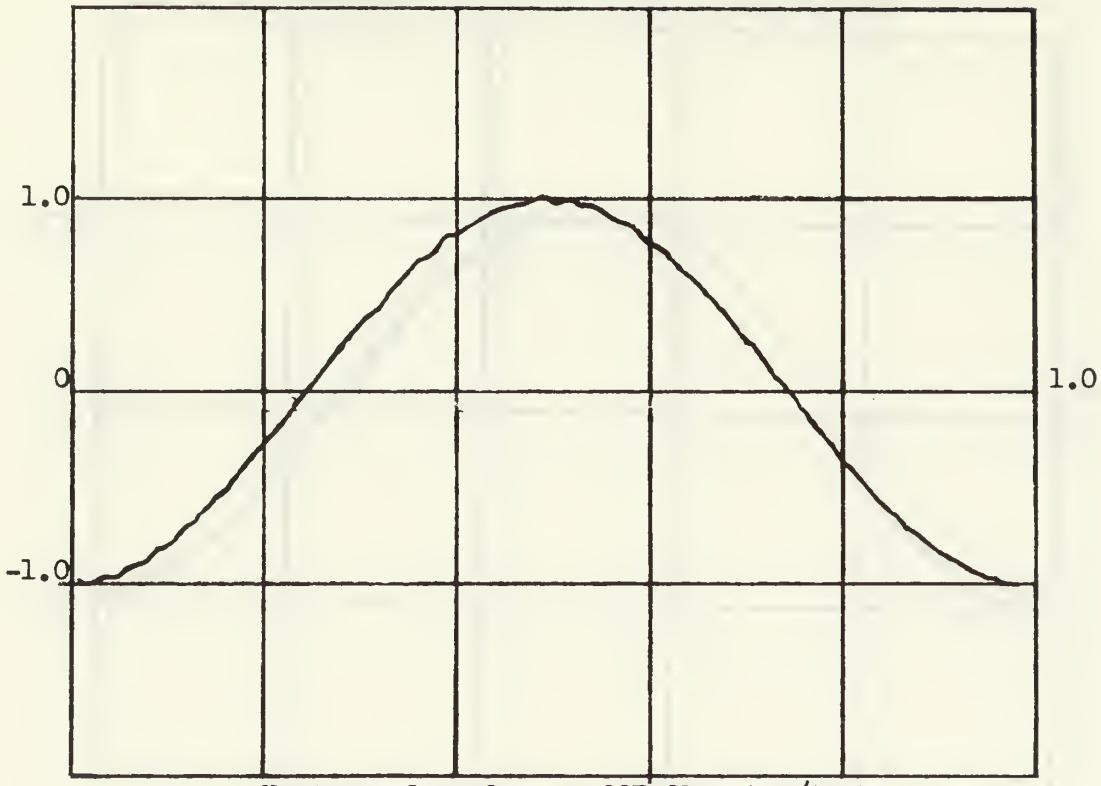
Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t = 3.50 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



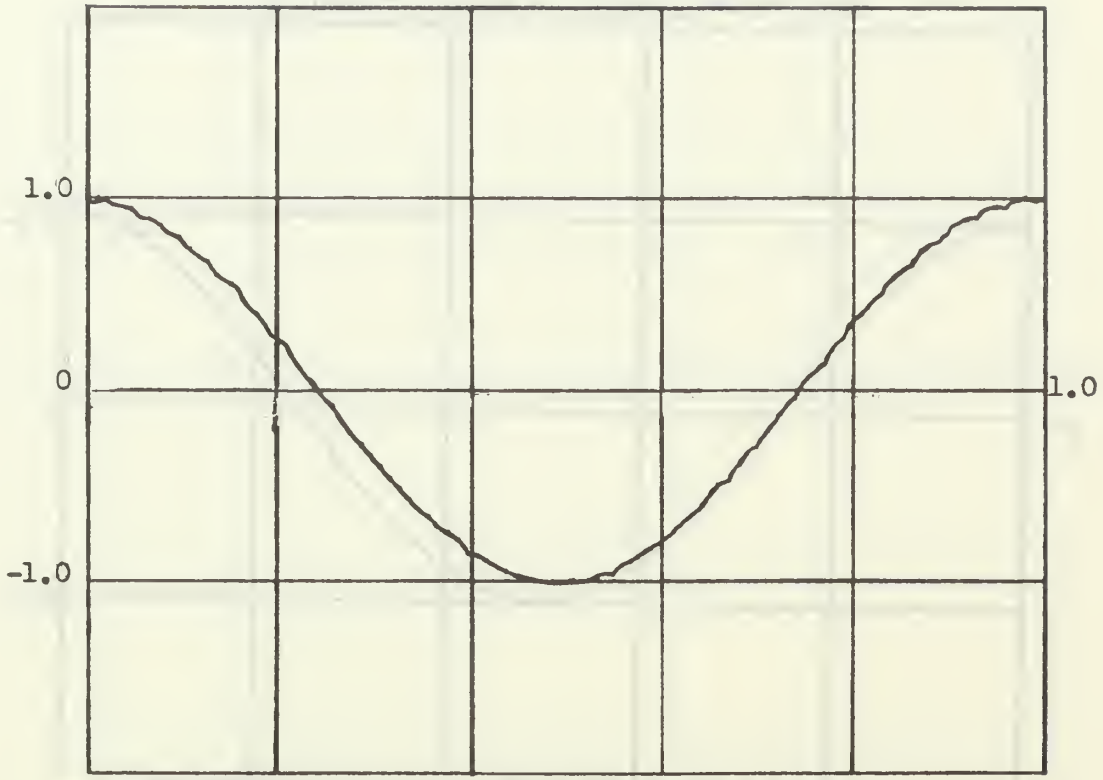
Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t = 4.00 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



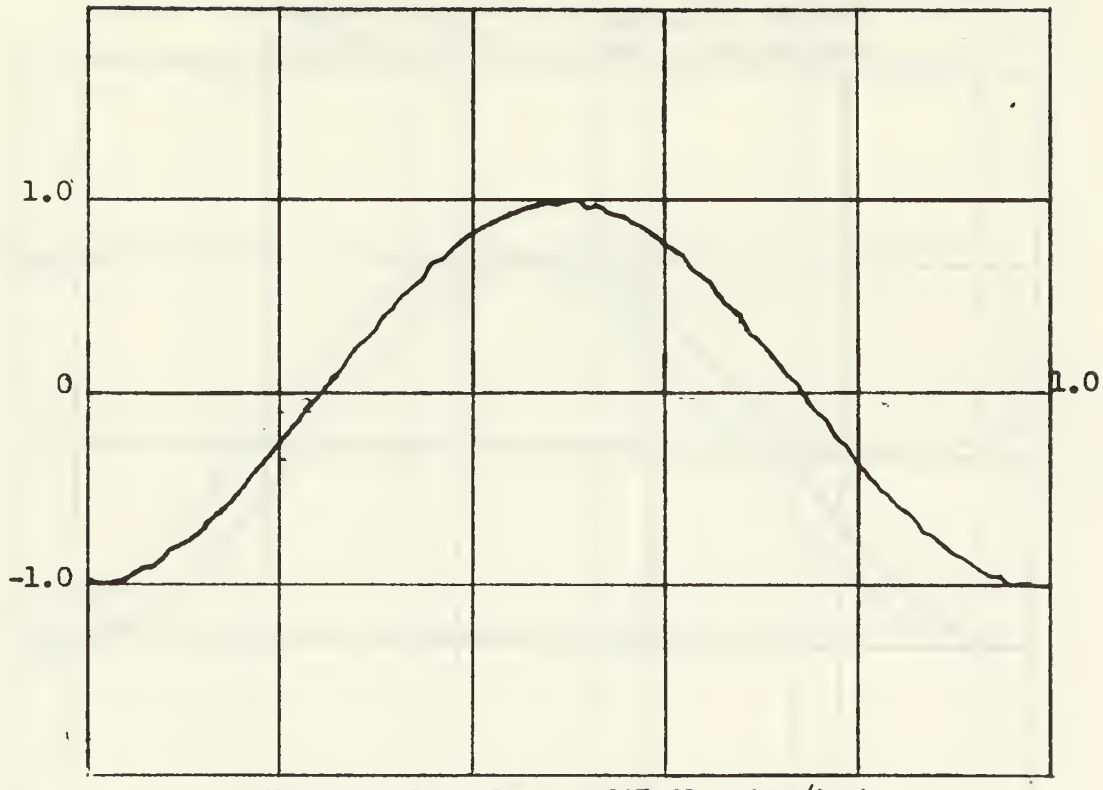
Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t = 4.50 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



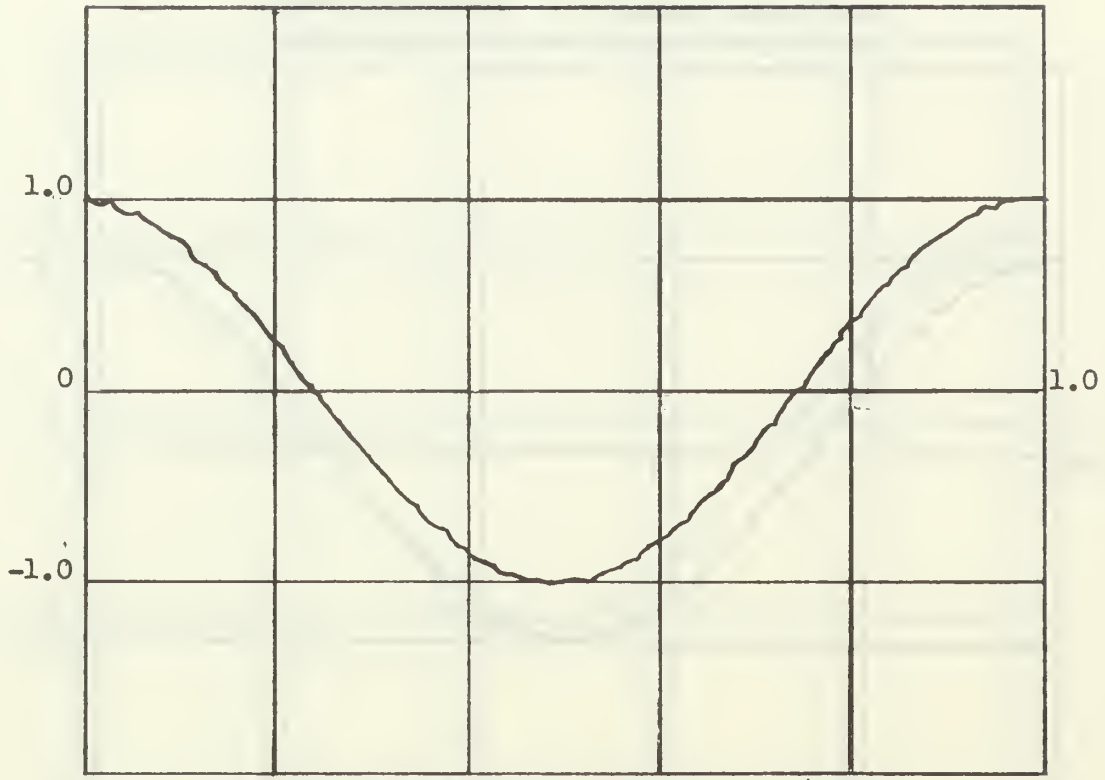
Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t = 5.00 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



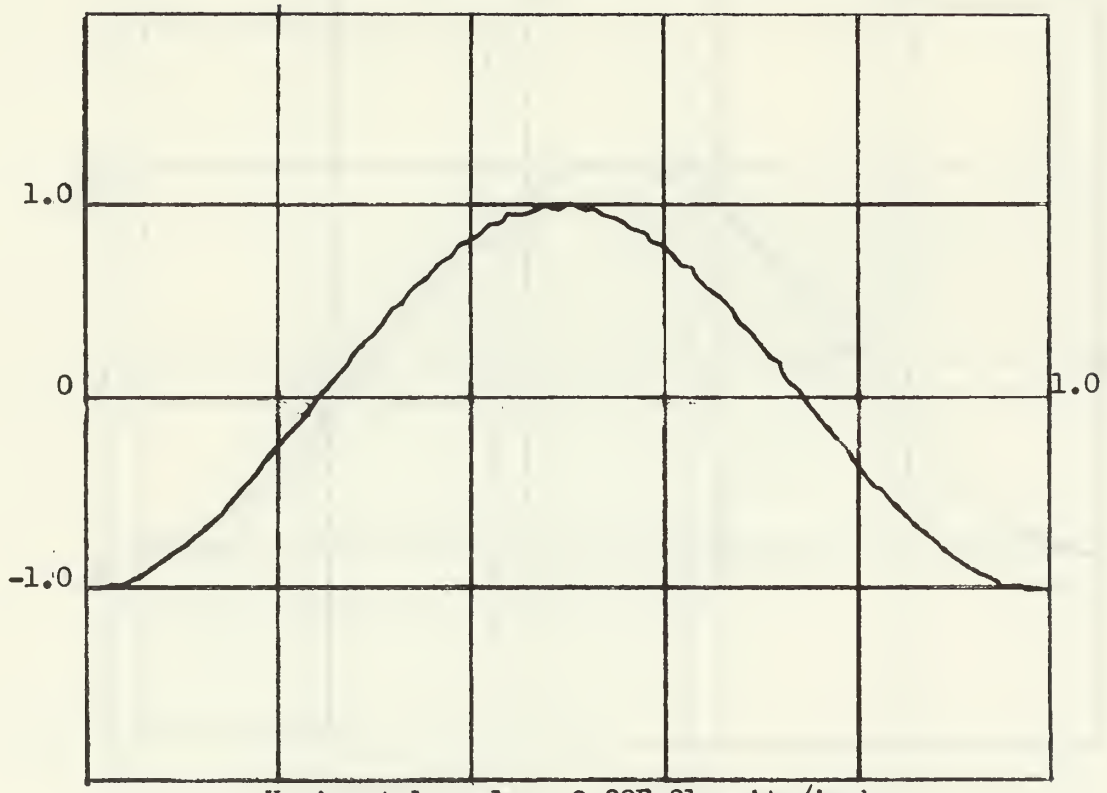
Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t = 5.50 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



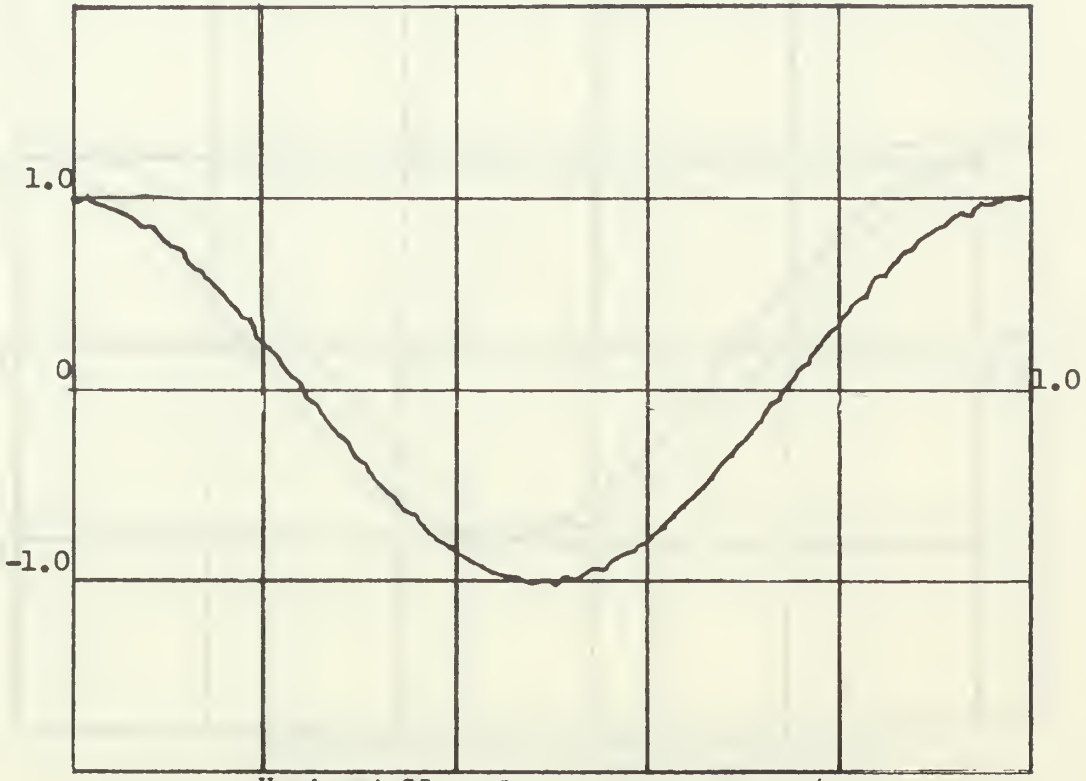
Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t = 6.00 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



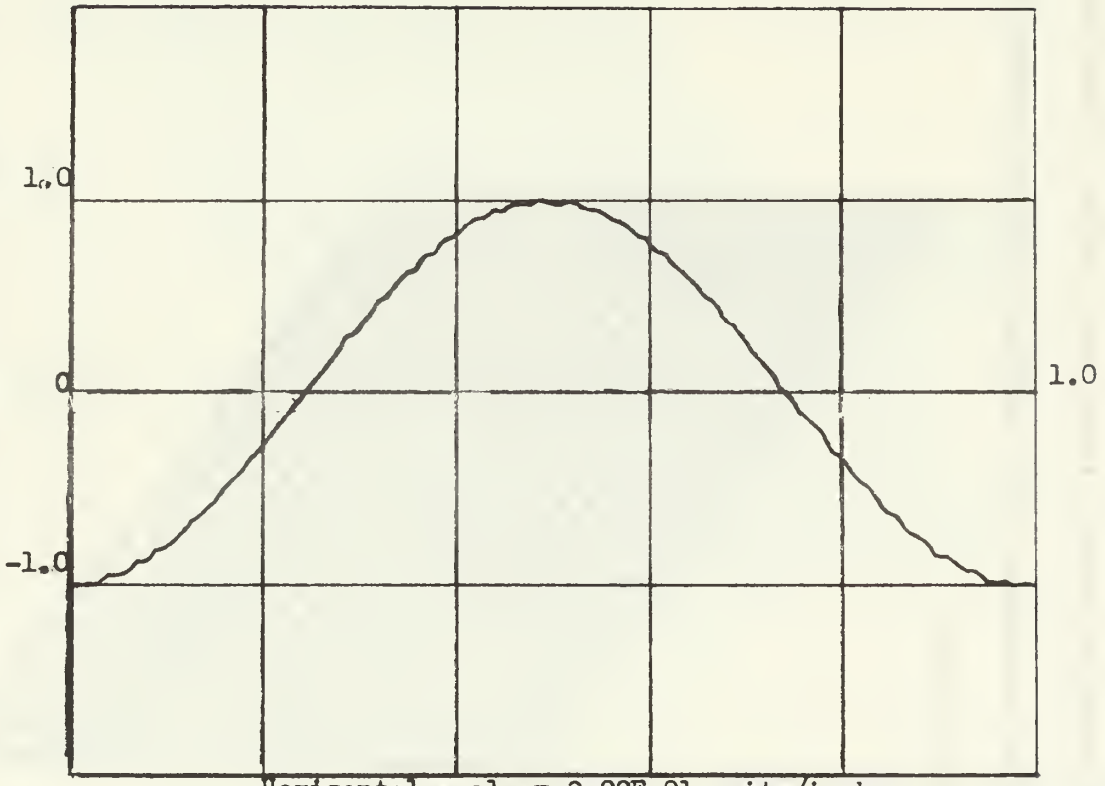
Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t = 6.50 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t = 7.00 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01



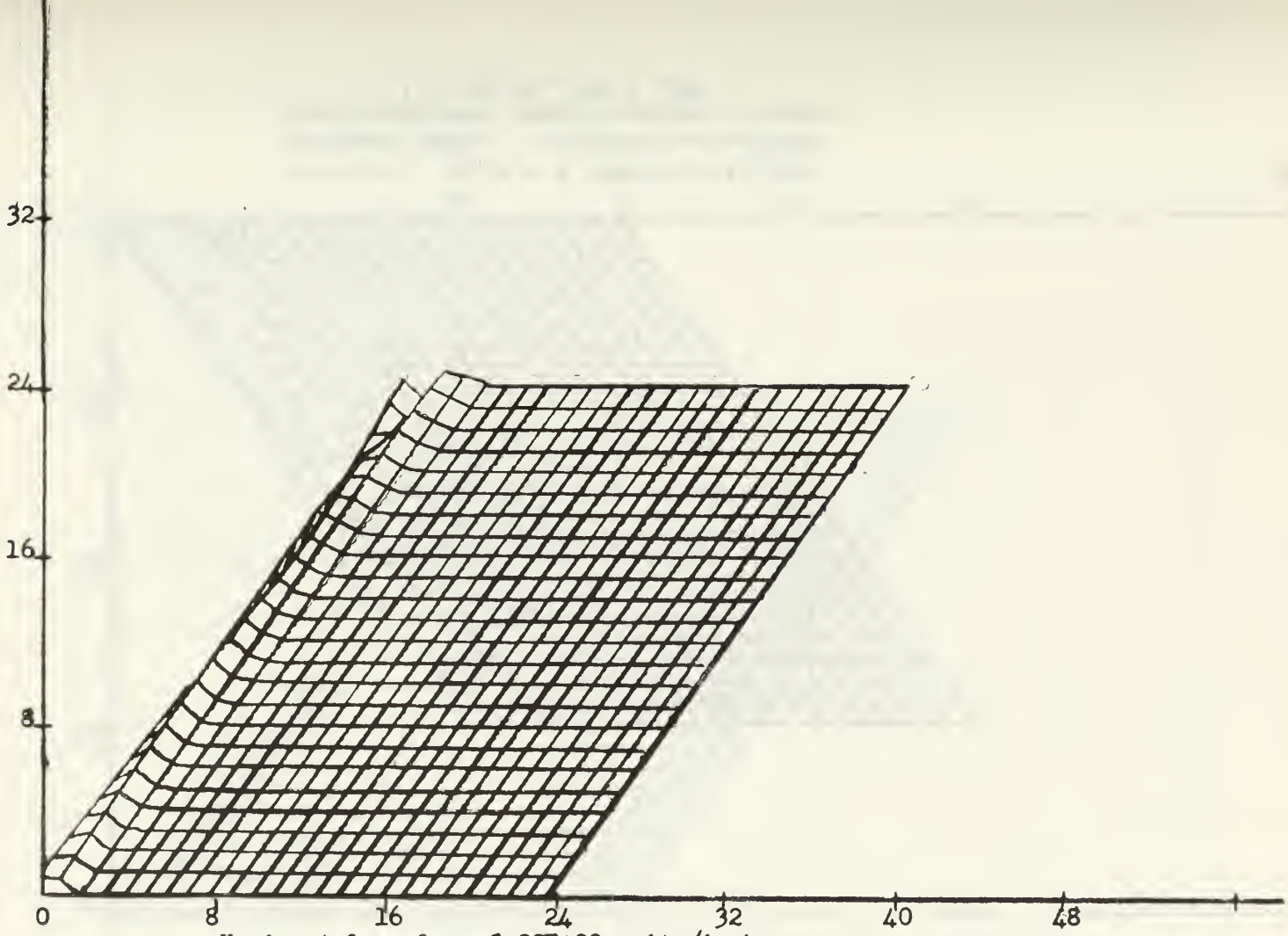
Horizontal scale = 2.00E-01 units/inch
Vertical scale = 1.00E+00 units/inch

Spacial pressure distribution at t = 7.50 sec. of a matched 1-dimensional domain to a unit step input. Length=1.0 unit. dx=0.01

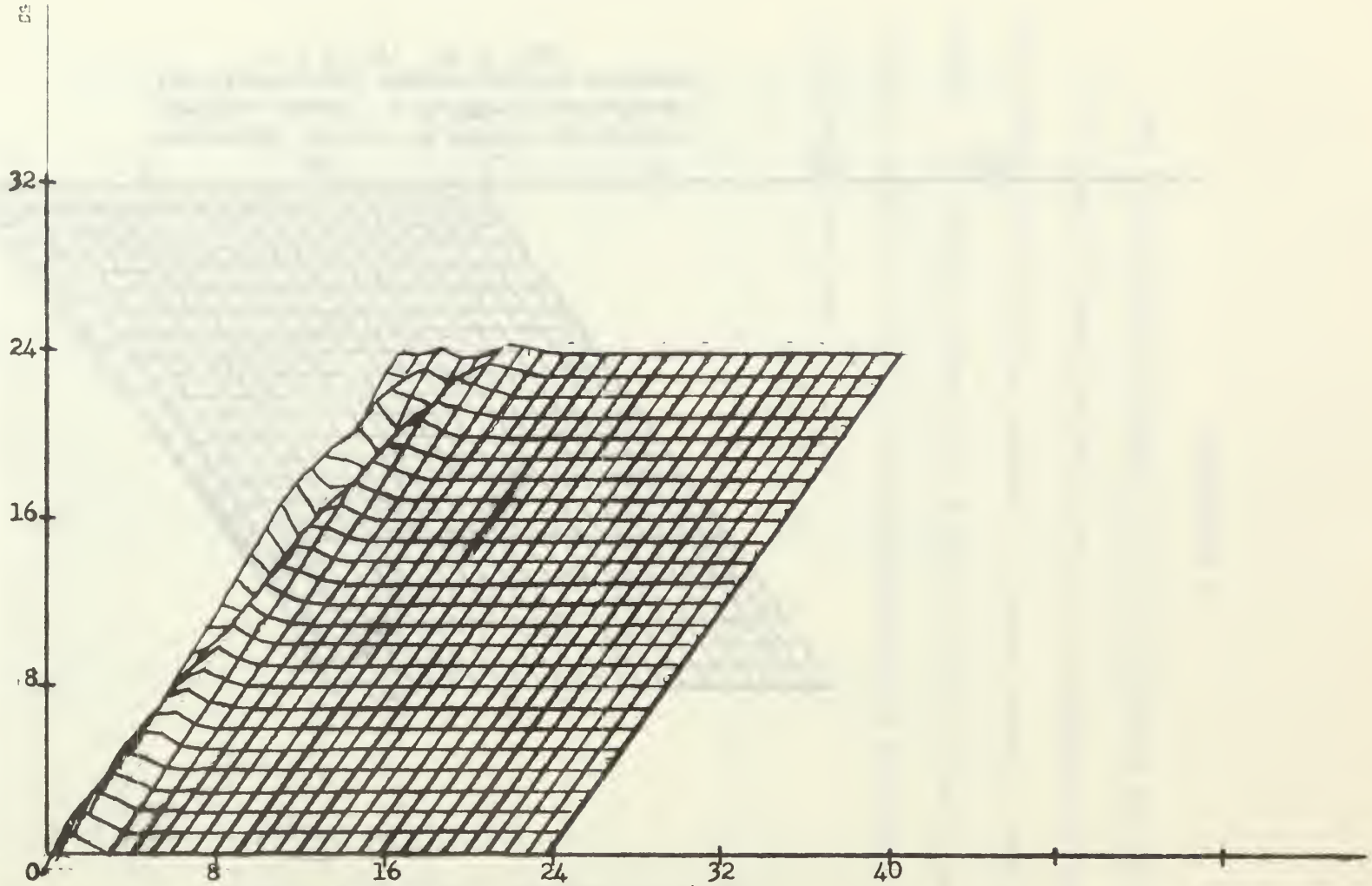
APPENDIX D

TWO-DIMENSIONAL VARIABLE-VELOCITY PROBLEM SOLUTION

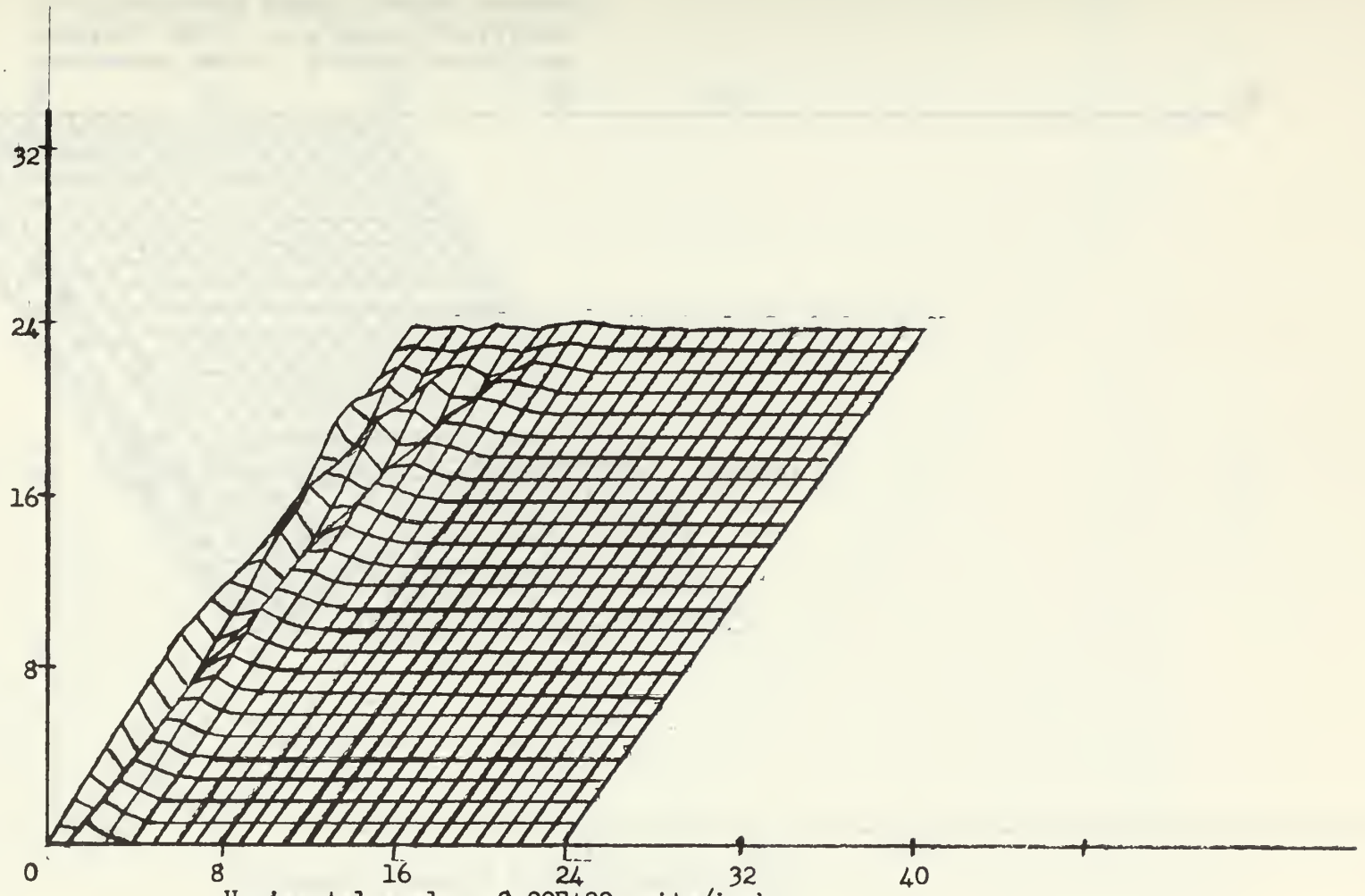
This appendix shows the two-dimensional pressure distribution for a line source initial condition applied to a medium having a variable-velocity profile. The example shown is for a case in which a linear increasing velocity profile is present. This condition would be present in seawater as the depth increases.



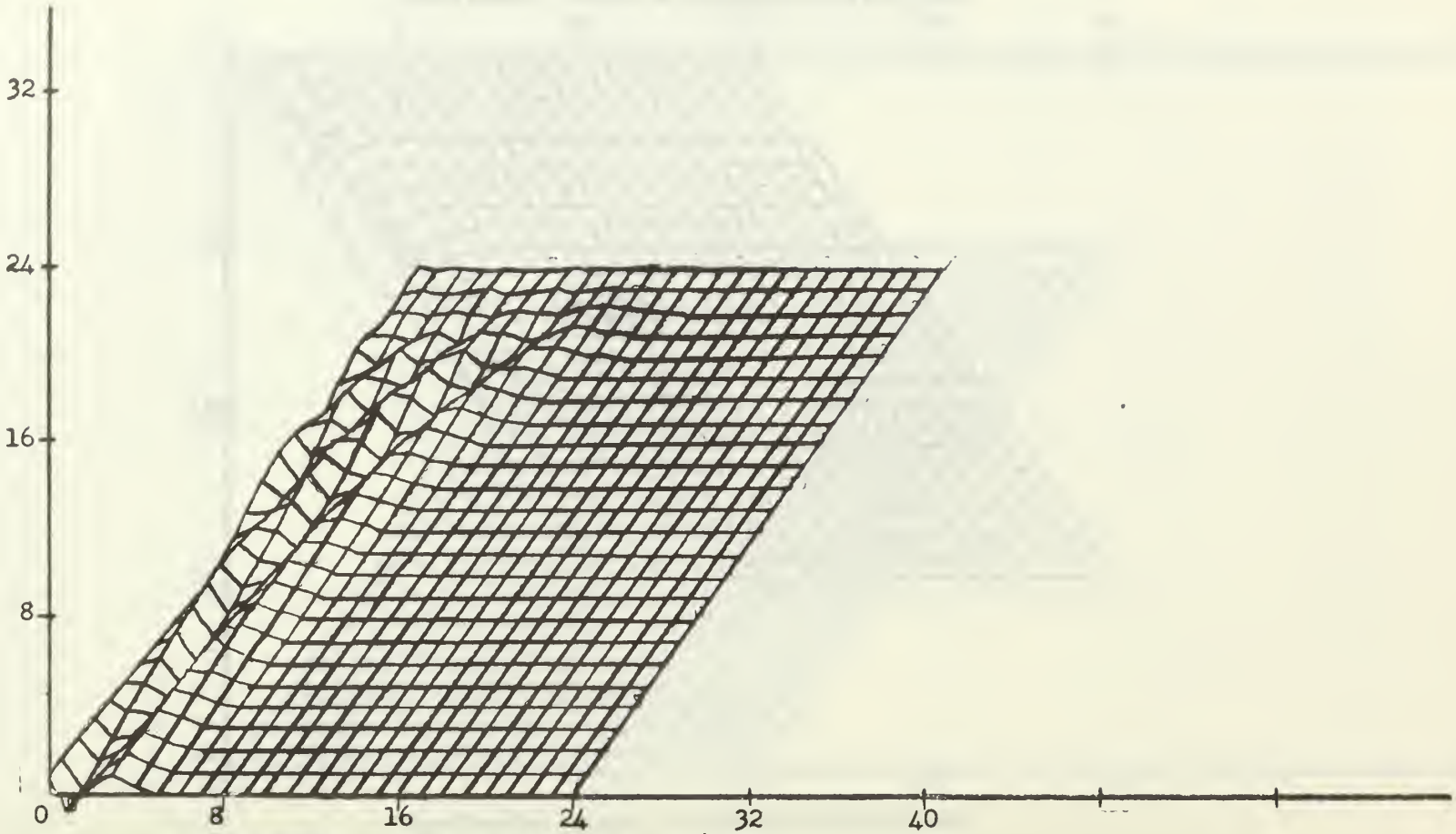
Horizontal scale = 8.00E+00 units/inch
Vertical scale = 8.00E+00 units/inch
Two-dimensional domain impulse response
t = 80 DT. DT = .001



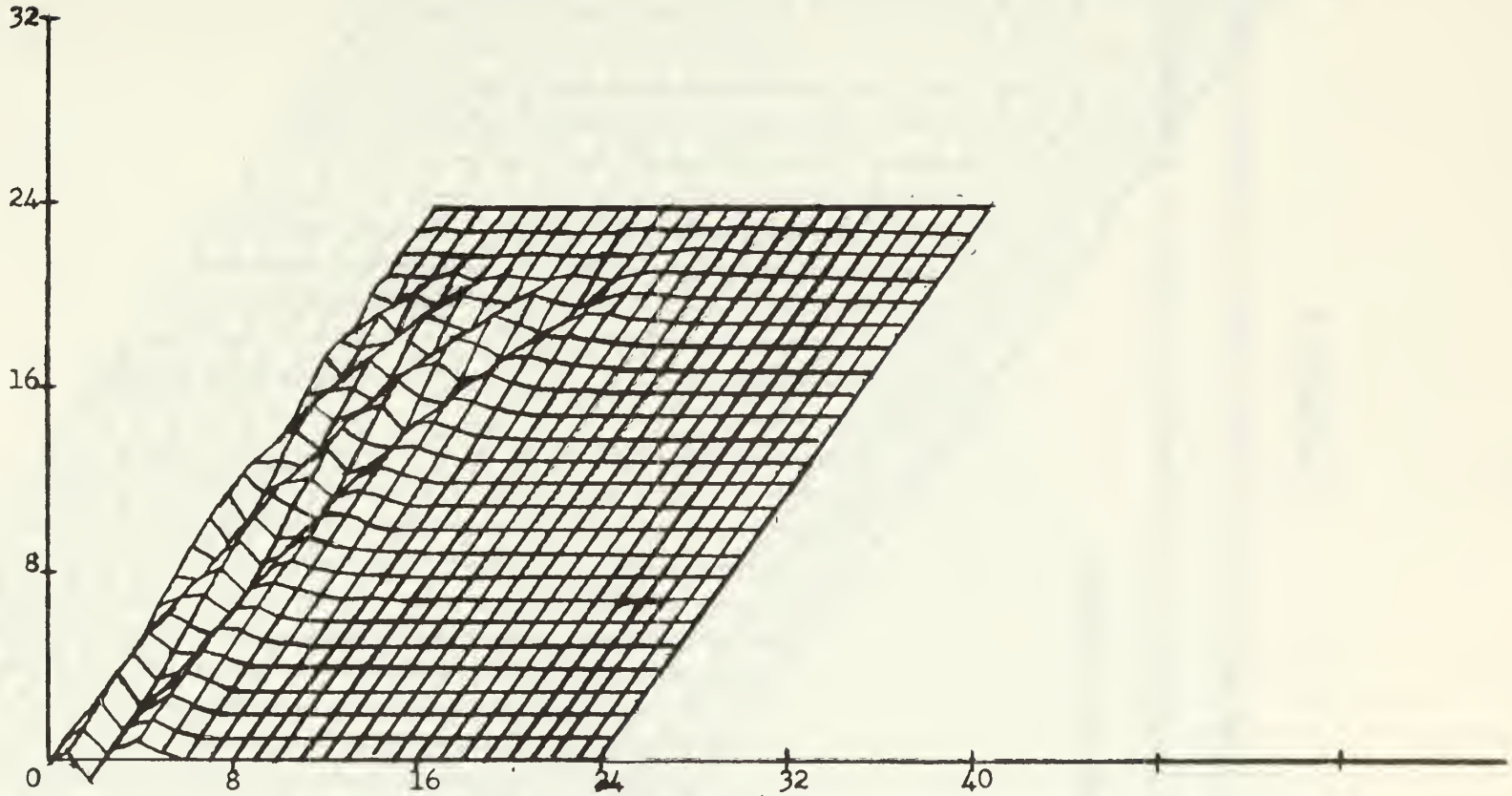
Horizontal scale = 8.00E+00 units/inch
Vertical scale = 8.00E+00 units/inch
Two-dimensional domain impulse response
t = 160 DT. DT = .001



Horizontal scale = 8.00E+00 units/inch
Vertical scale = 8.00E+00 units/inch
Two-dimensional domain impulse response
t = 240 DT. DT = .001



Horizontal scale = 8.00E+00 units/inch
Vertical scale = 8.00E+00 units/inch
Two-dimensional domain impulse response
t = 320 DT. DT = .001

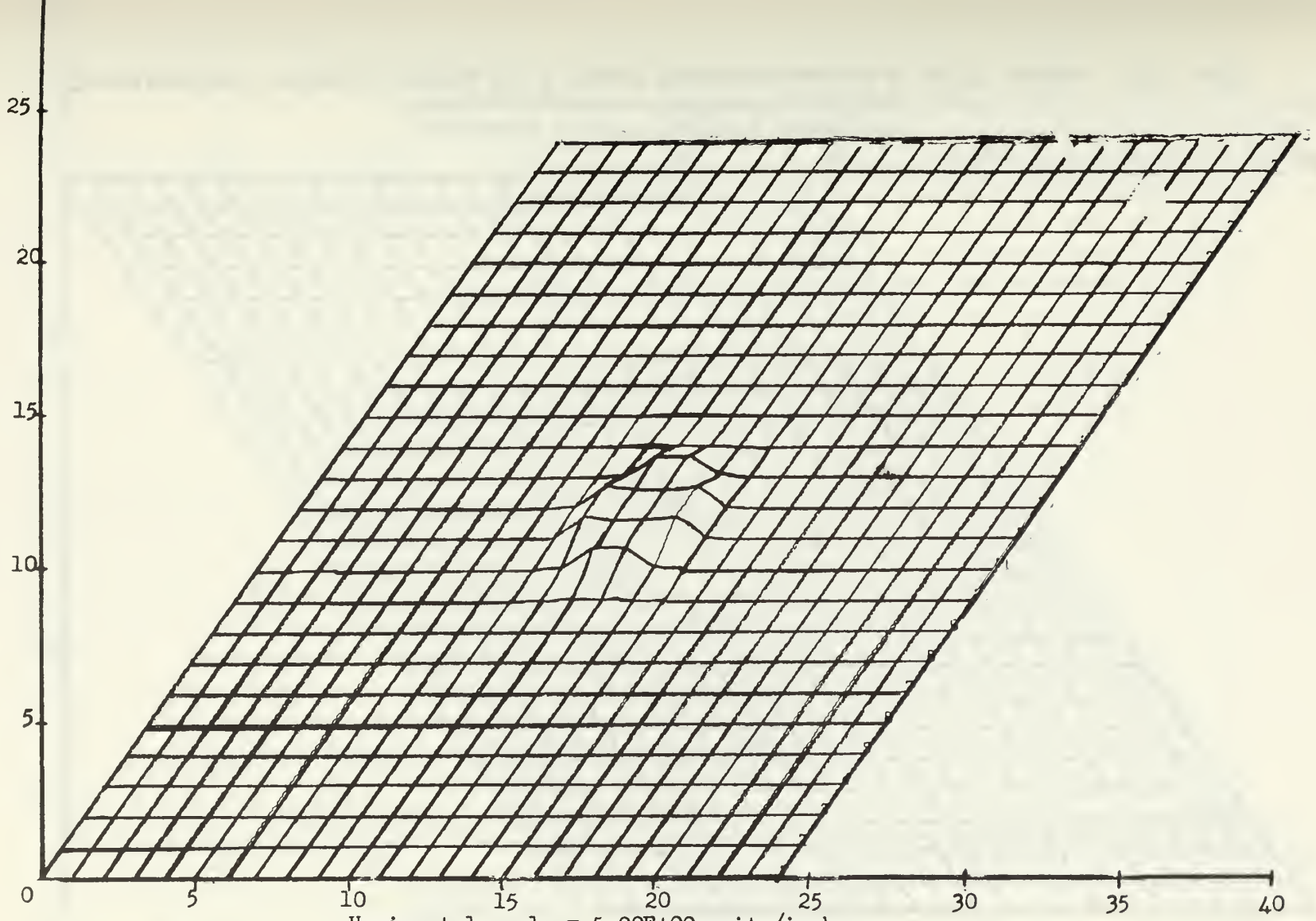


Horizontal scale = 8.00E+00 units/inch
Vertical scale = 8.00E+00 units/inch
Two-dimensional domain impulse response
t = 400 DT. DT = .001

APPENDIX E

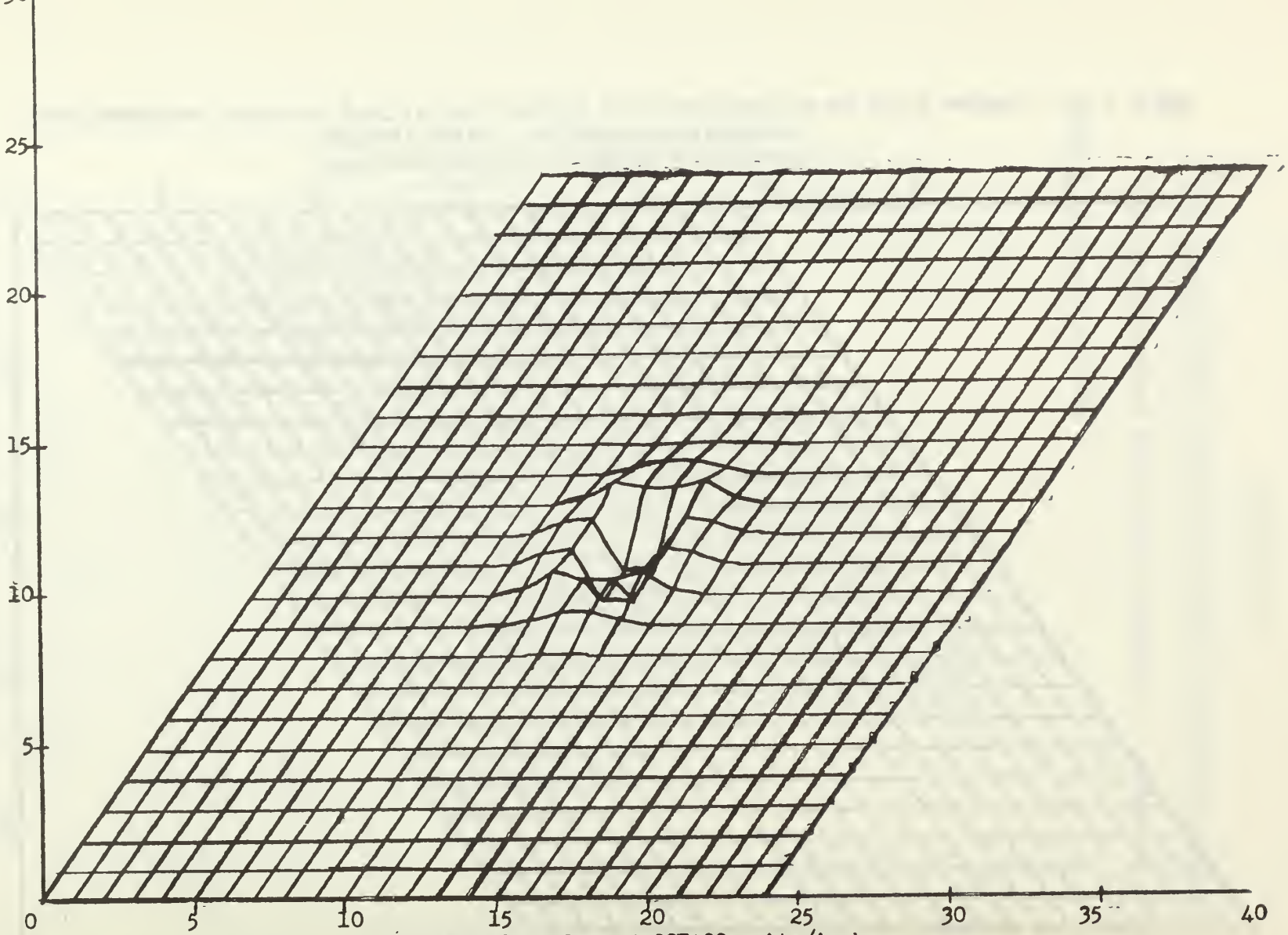
TWO-DIMENSIONAL NON-ZERO INITIAL CONDITION SOLUTION

This appendix shows the transient solution to a deformed two-dimensional membrane problem.



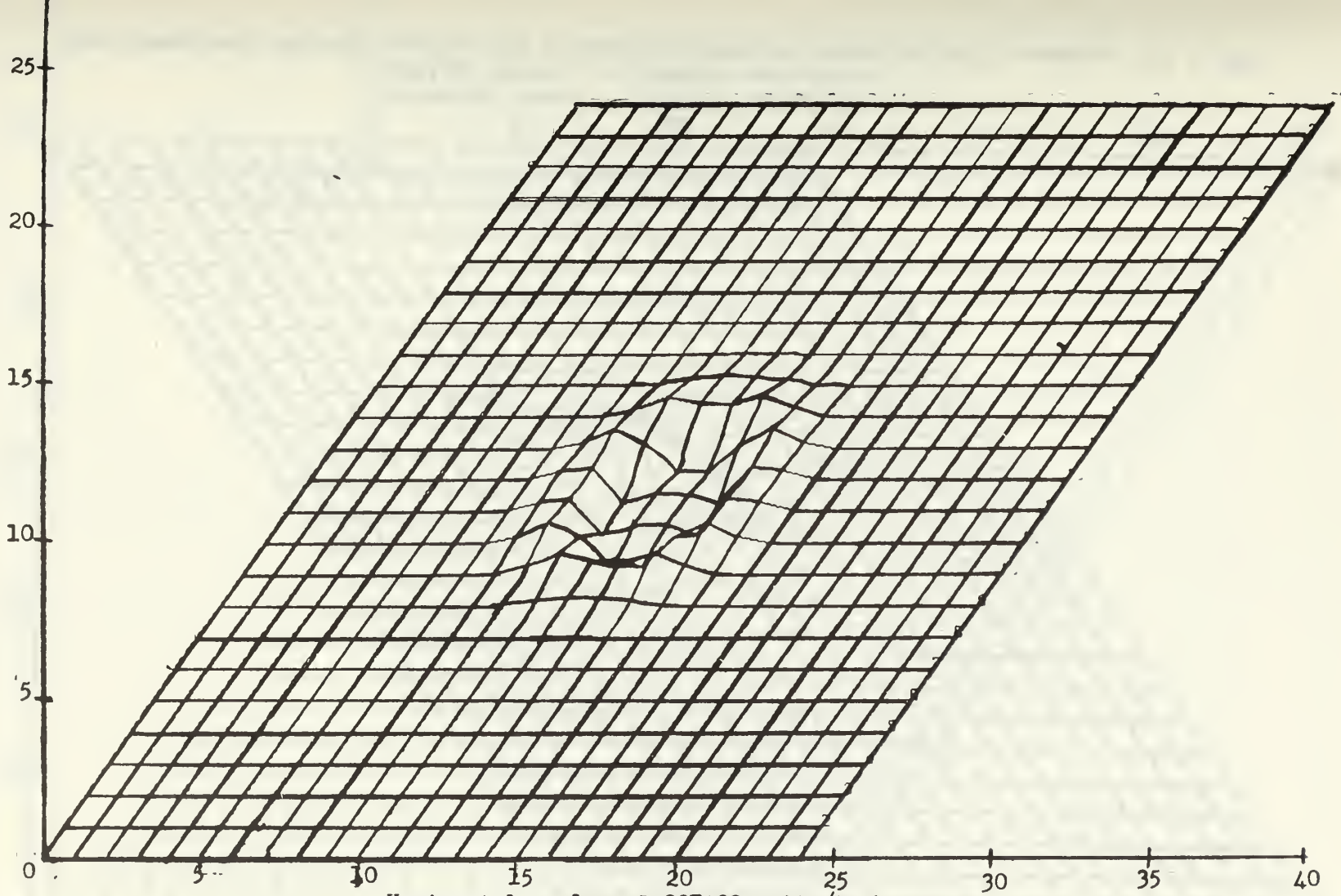
Horizontal scale = 5.00E+00 units/inch
Vertical scale = 5.00E+00 units/inch

Two-dimensional velocity profile for a highly dispersive medium at 20 DT seconds. DT = 0.001



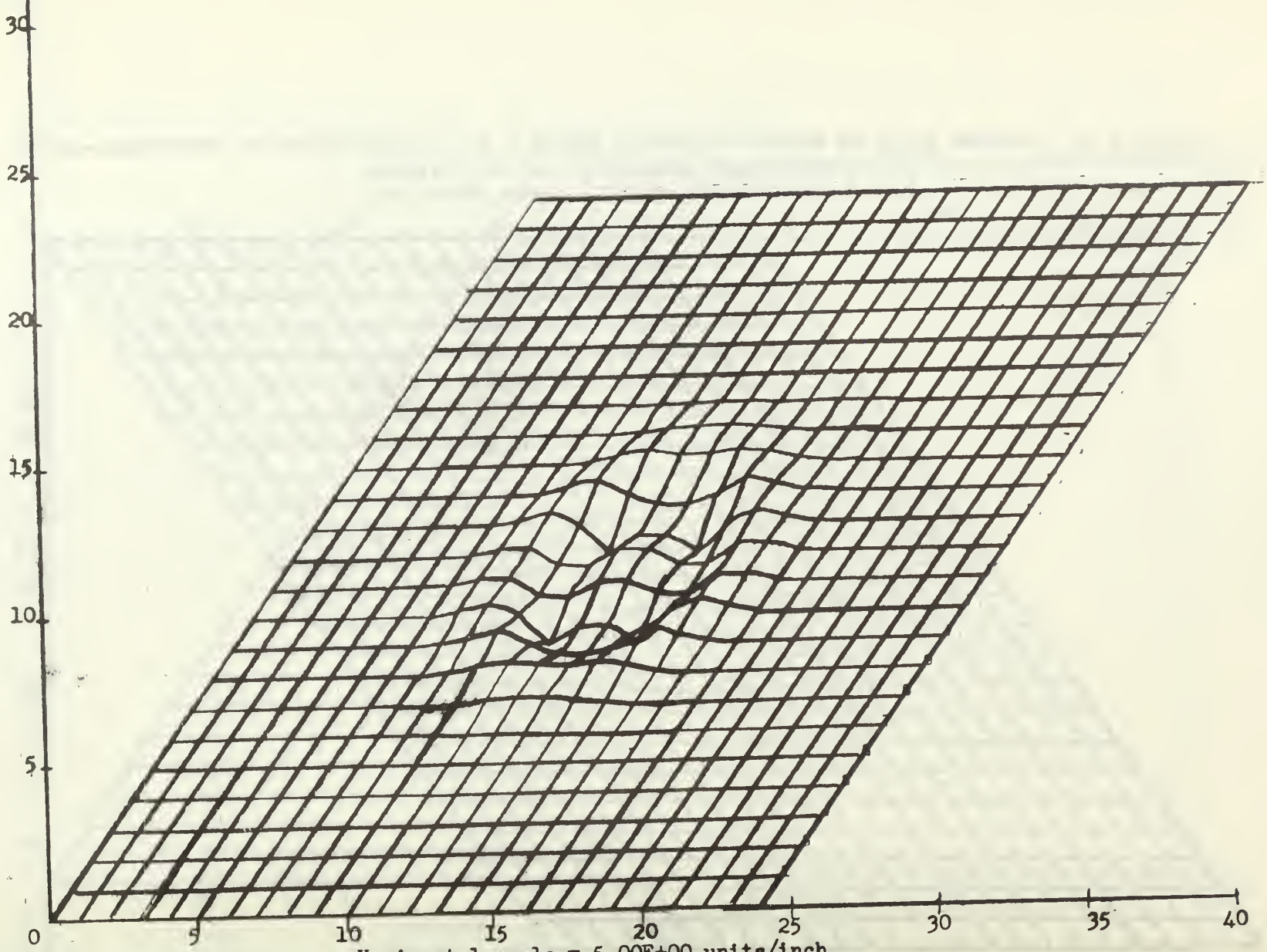
Horizontal scale = 5.00E+00 units/inch
Vertical scale = 5.00E+00 units/inch

Two-dimensional velocity profile for a highly dispersive medium at 40 DT seconds. DT = 0.001



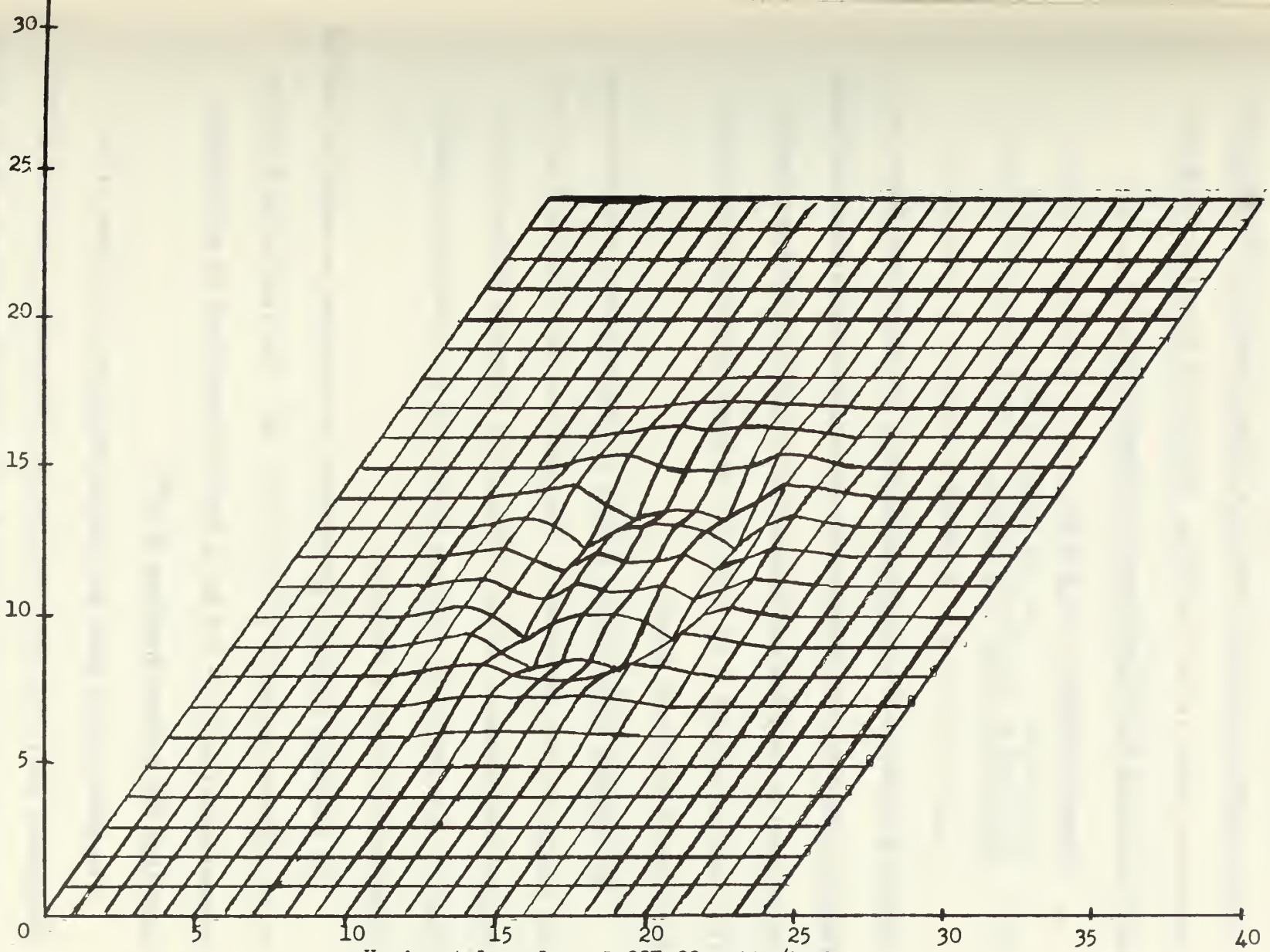
Horizontal scale = 5.00E+00 units/inch
Vertical scale = 5.00E+00 units/inch

Two-dimensional velocity profile for a highly dispersive medium at 60 DT seconds. DT = 0.001



Horizontal scale = 5.00E+00 units/inch
Vertical scale = 5.00E+00 units/inch

Two-dimensional velocity profile for a highly dispersive medium at 80 DT seconds. DT = .001



Horizontal scale = 5.00E+00 units/inch
Vertical scale = 5.00E+00 units/inch

Two-dimensional velocity profile for a highly dispersive medium at 100 DT seconds. DT = 0.001

APPENDIX F

The graphical results of running a Widrow adaptive filter simulation program are shown in this appendix. The desired signal to which the array responded had the following characteristics:

center frequency	= 500 Hz
band-width	= 200 Hz and 0 Hz
desired look angle	= 45 degrees
power signal input	= 1.0 watt

The noise signal is of standard deviation 0.707 and is provided by the random number program GUASS; thus for wideband signals the total noise power input to the array may exceed that of the signal before filtering. The noise is incident at an angle of 0 degrees relative to the normal of the plane of the linear array.

The filtering action occurs in both in frequency response selectivity and spacial angular selectivity. The graphs on the following pages are grouped into frequency response graphs and receiving pattern graphs. There are four frequency response graphs and four receiving patterns shown for each time of adaption.

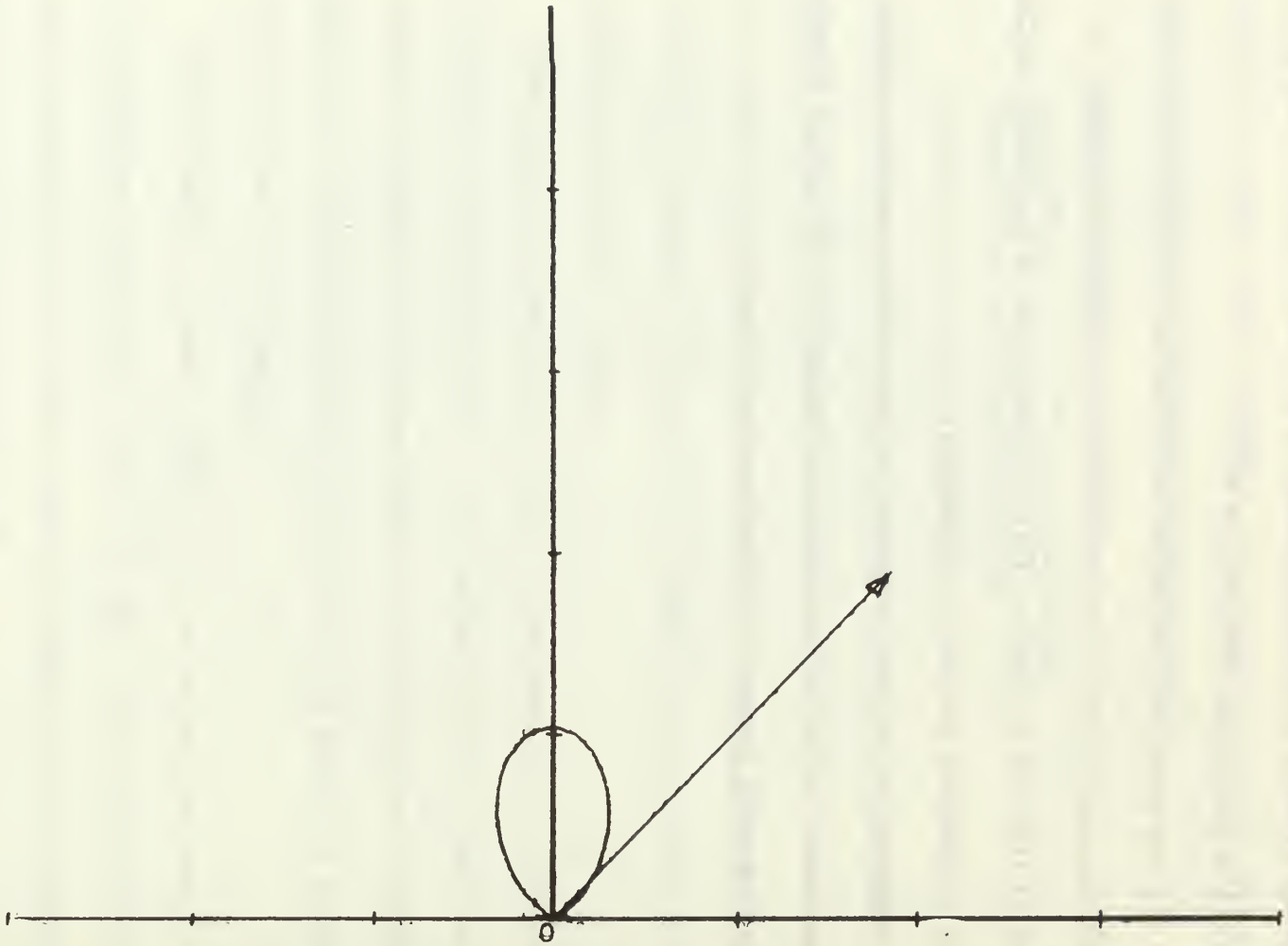
The frequency response graphs show the response as would be observed for the different angles of 0°, 30°, 45°, 90°. The results show that the desired signal of 500 ± 100 Hz is strongly attenuated for all directions other than the desired direction of 45°.

The pattern plots show the spacial response of the filter at the frequencies 100, 200, 500, and 1000 Hz. The results show that these is greatly reduced array pattern gain for the desired frequency of 500 ± 100 Hz

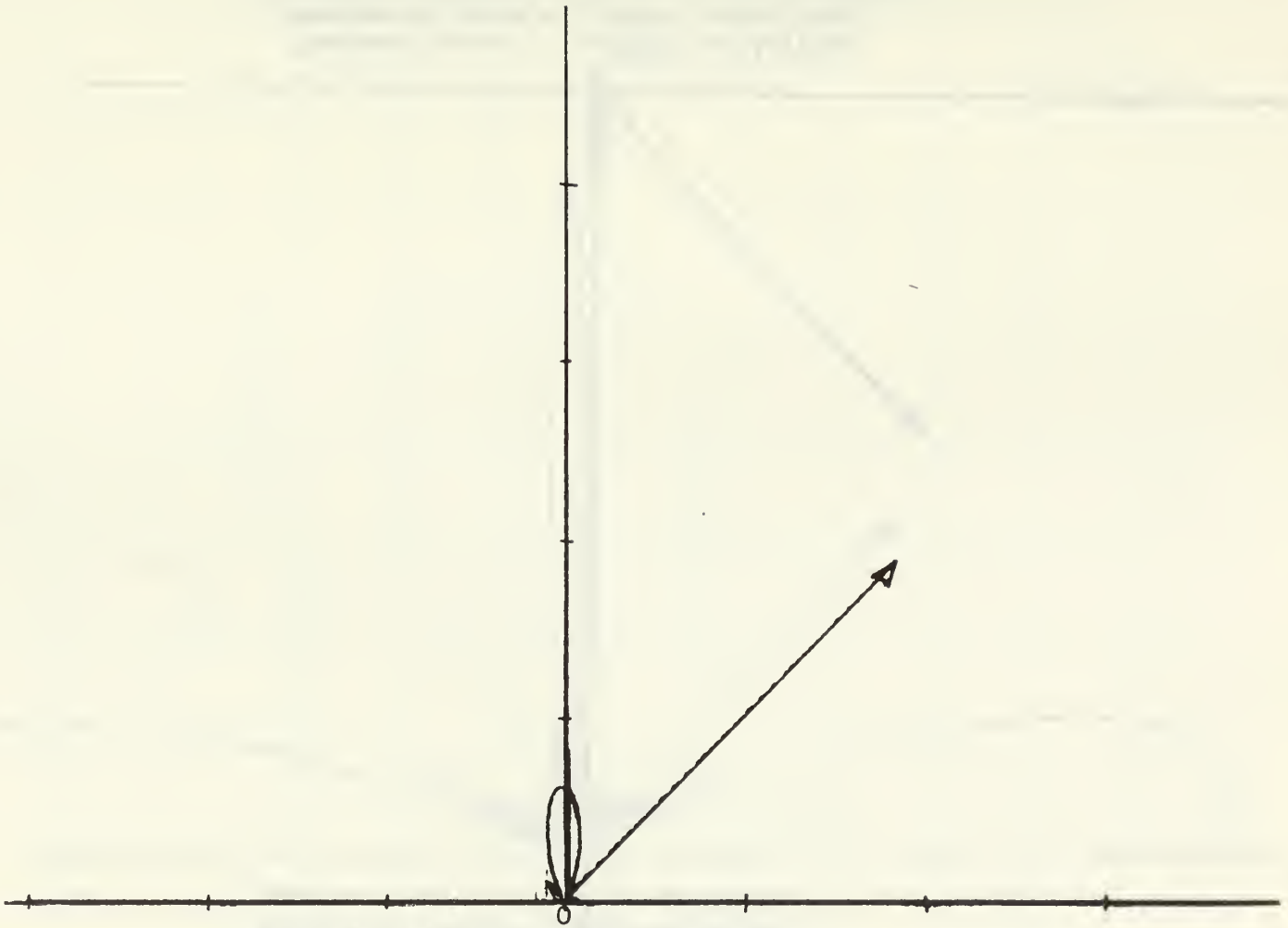
in all directions except the design direction of 45°. In addition, the response of array to the frequencies of 100 Hz and 200 Hz are greatly reduced in the design direction of 45°.

The one prominent exception to the desired behavior is that the filter has appreciable gain and frequency response for signals or noise at 1000 Hz or higher. This is correctable as mentioned in Chap. VI by appropriate low-pass prefiltering.

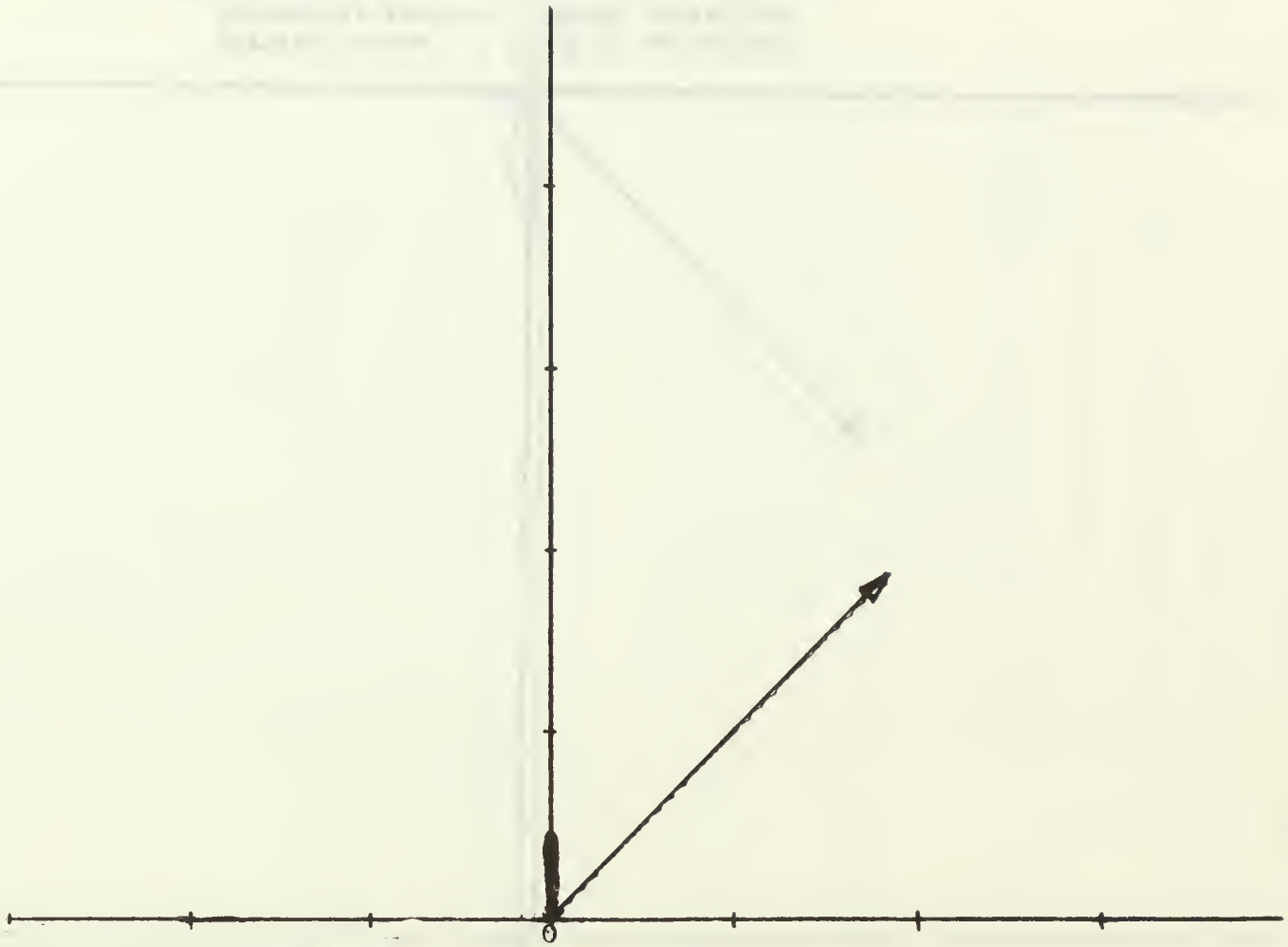
The results for zero bandwidth desired signals are very close to those shown for the wide-band signal except that the rate of adaption is considerably greater in the desired look direction.



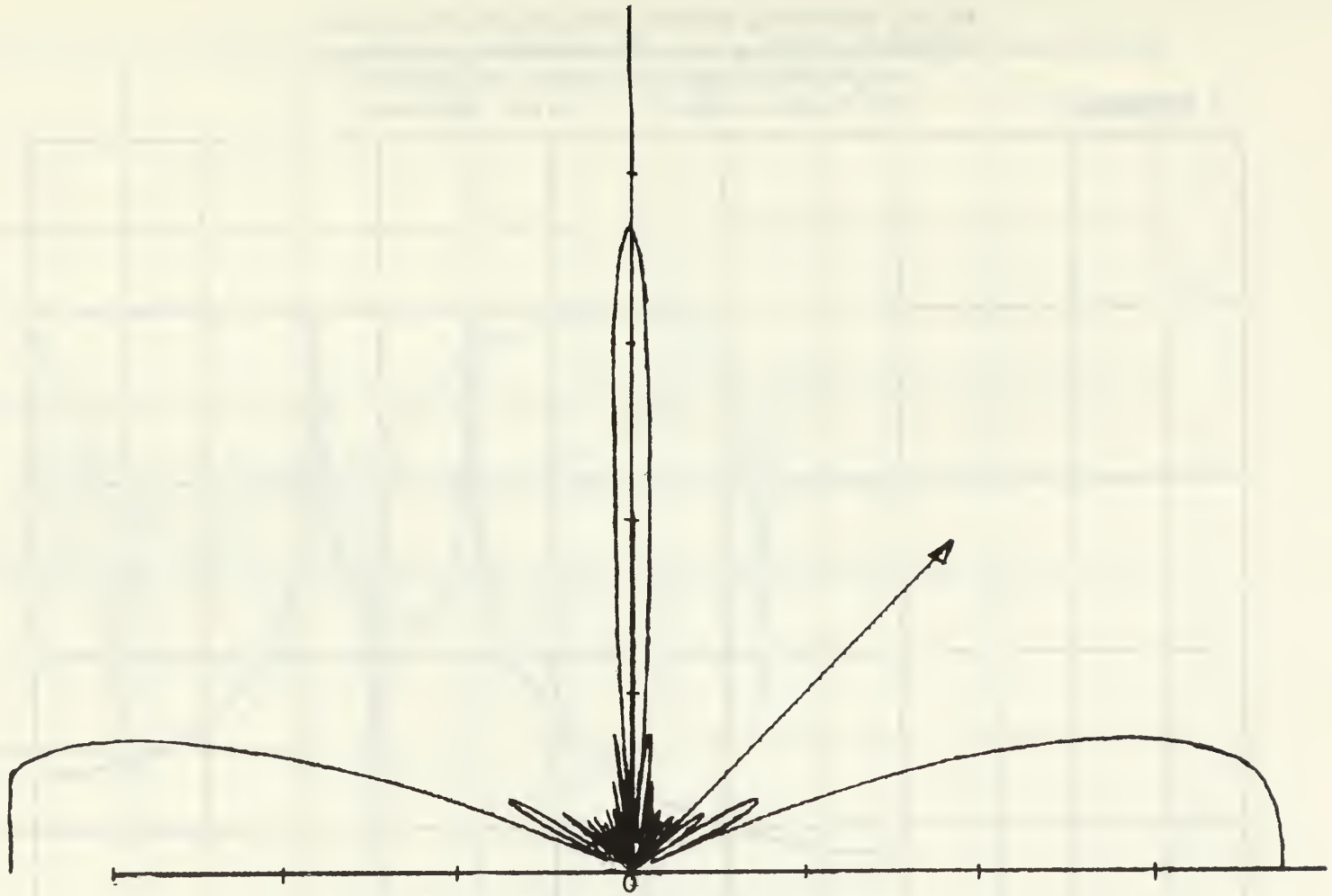
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 100 Hz after adapting
for 15 cycles of the desired frequency 500 Hz.



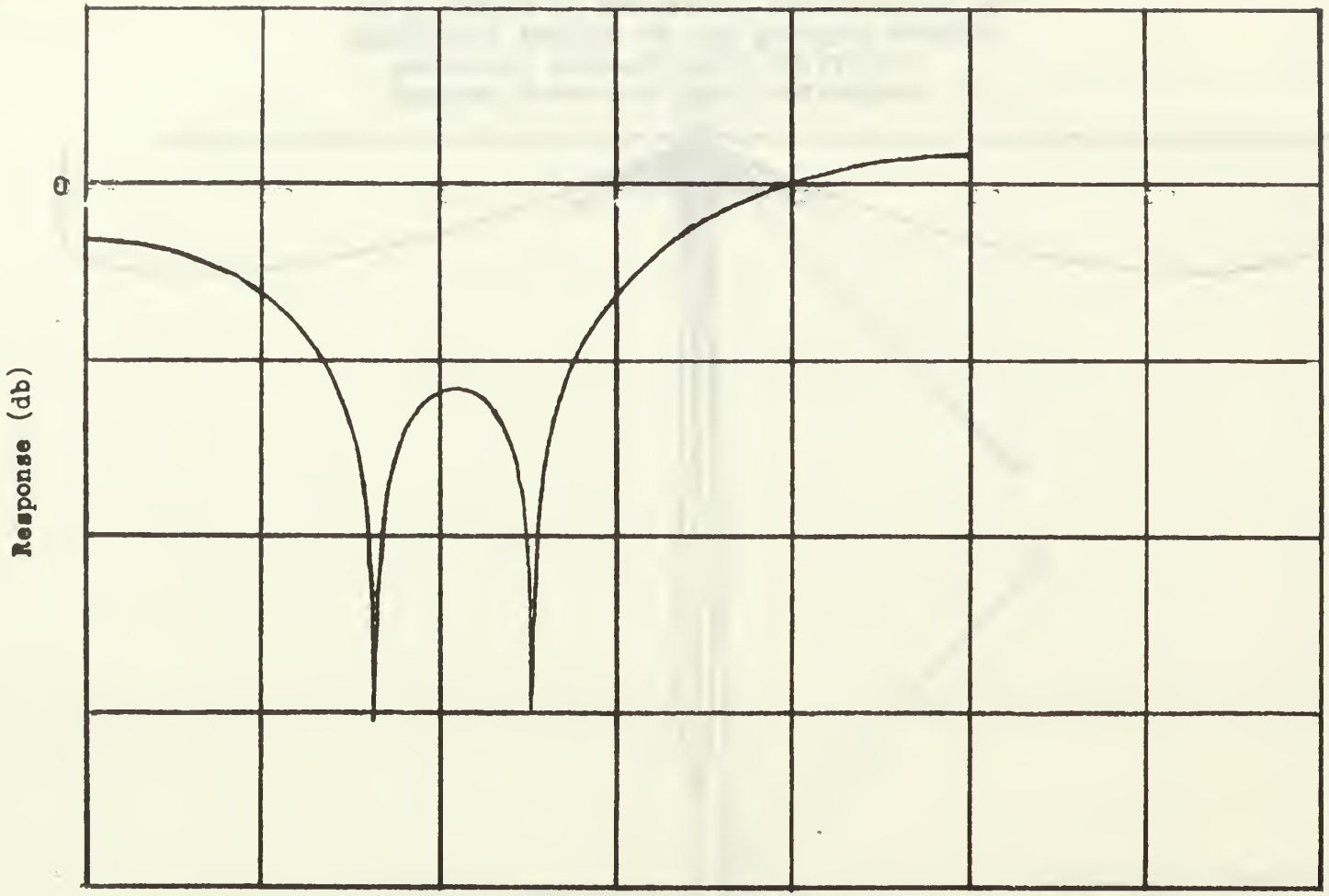
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 200 Hz after adapting
for 15 cycles of the desired frequency 500 Hz.



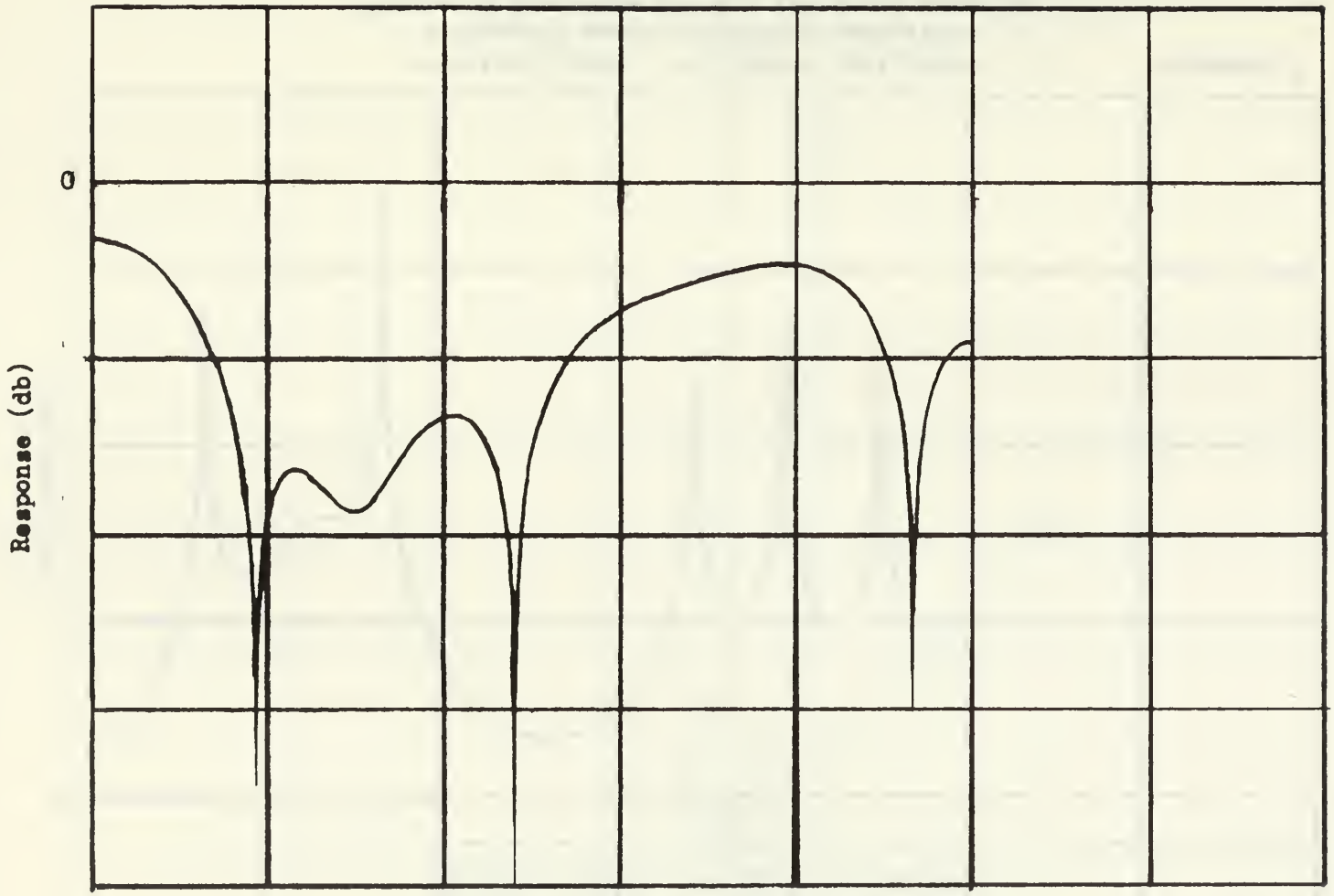
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 500 Hz after adapting for 15 cycles of the desired frequency 500 Hz.



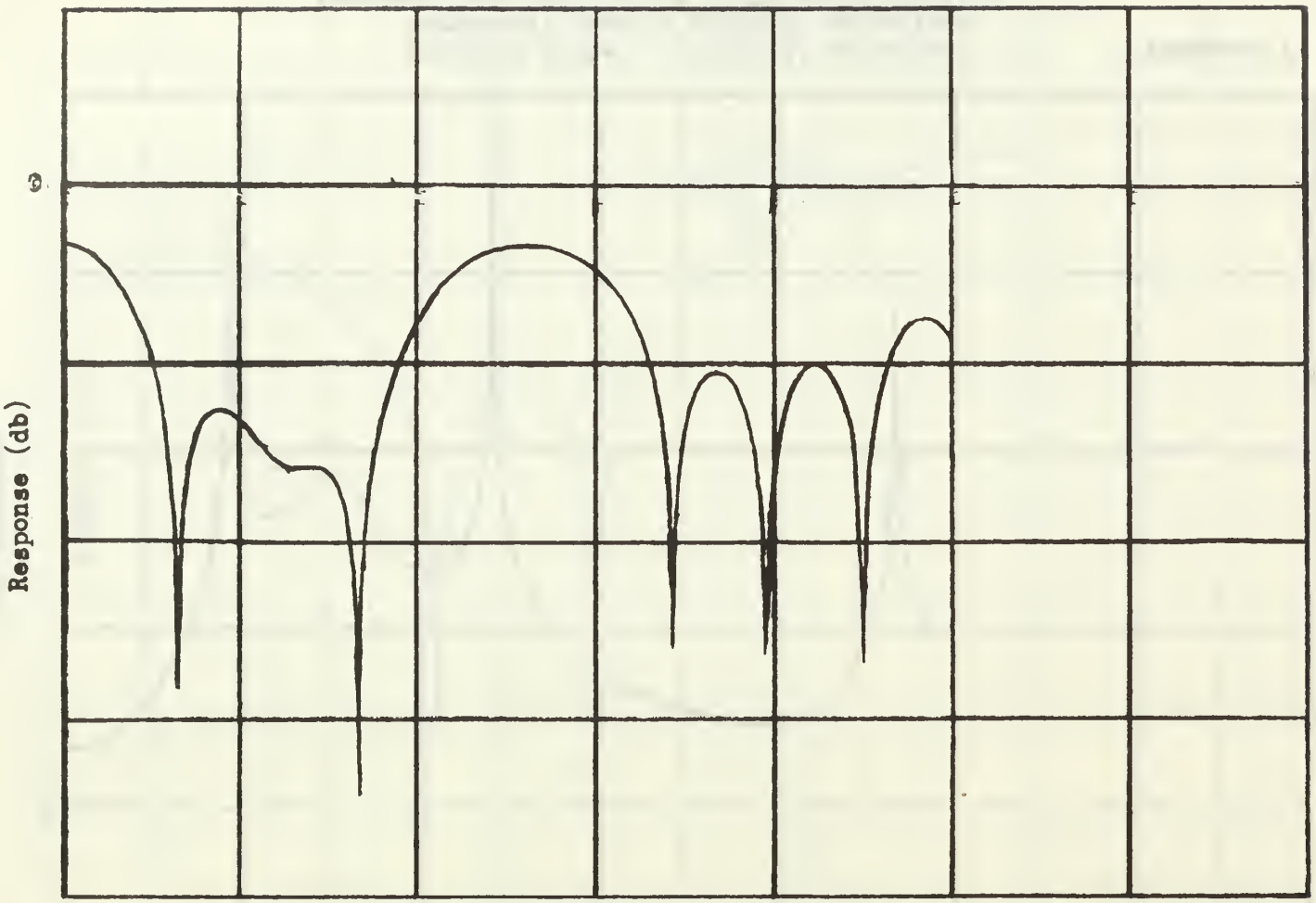
Vertical scale = 4.00E-01 units/inch
Horizontal scale=4.00E-01 units/inch
Directivity pattern at 1000 Hz after adapting
for 15 cycles of the desired frequency 500 Hz.



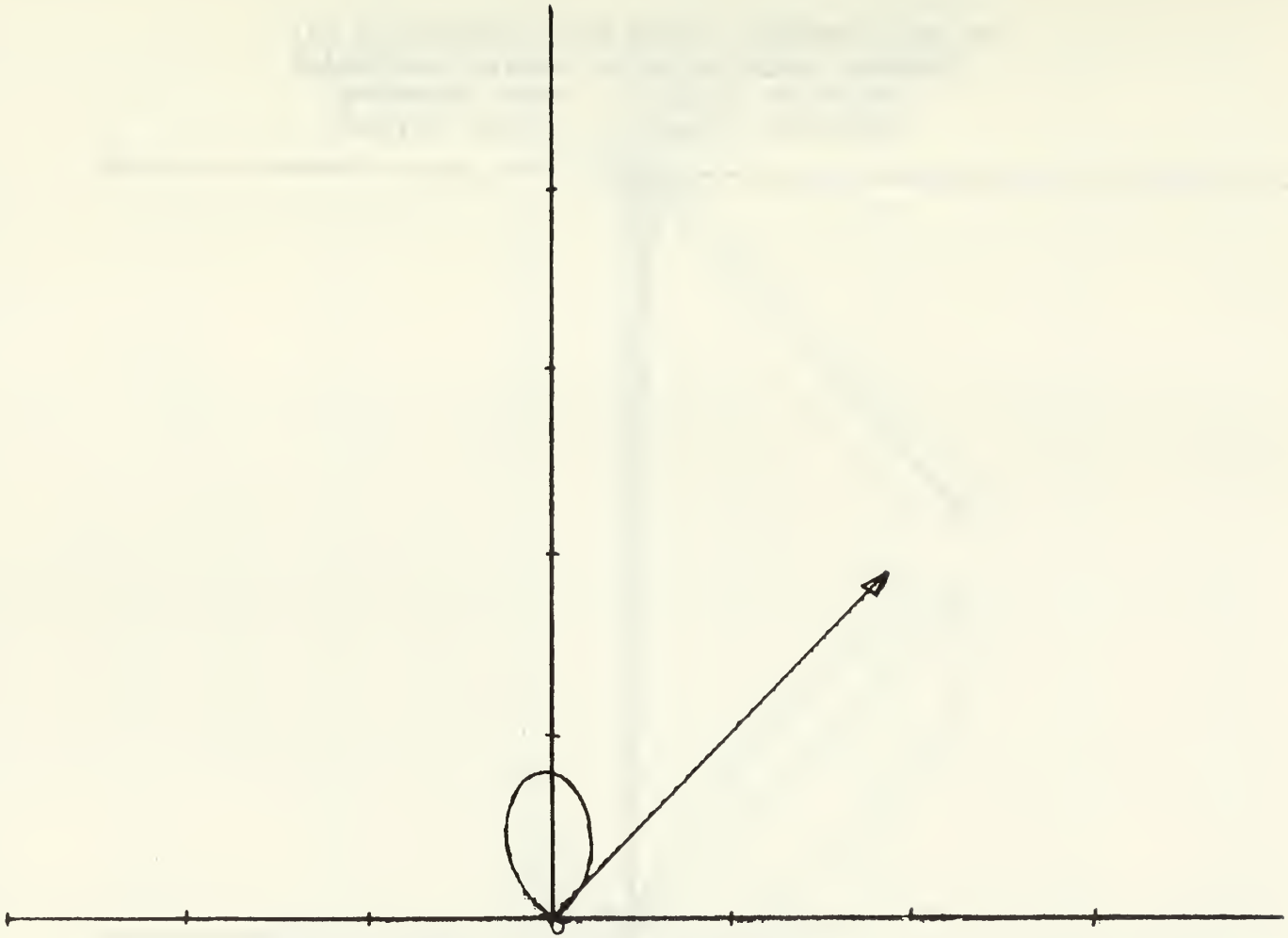
Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 0^{o}$ after adapting for 15 cycles of the desired frequency 500 Hz.



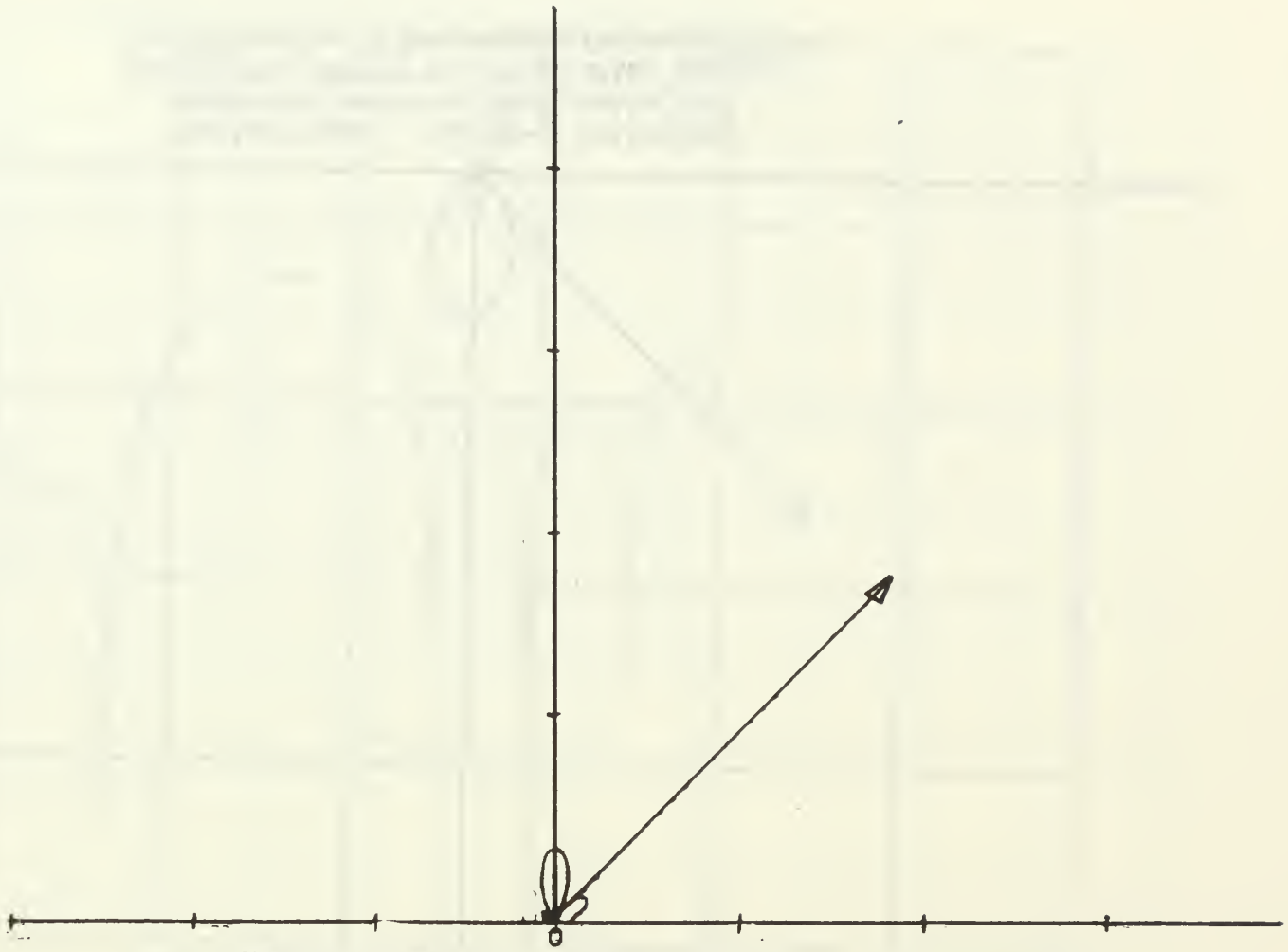
Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 30^{o}$ after adapting
for 15 cycles of the desired frequency 500 Hz.



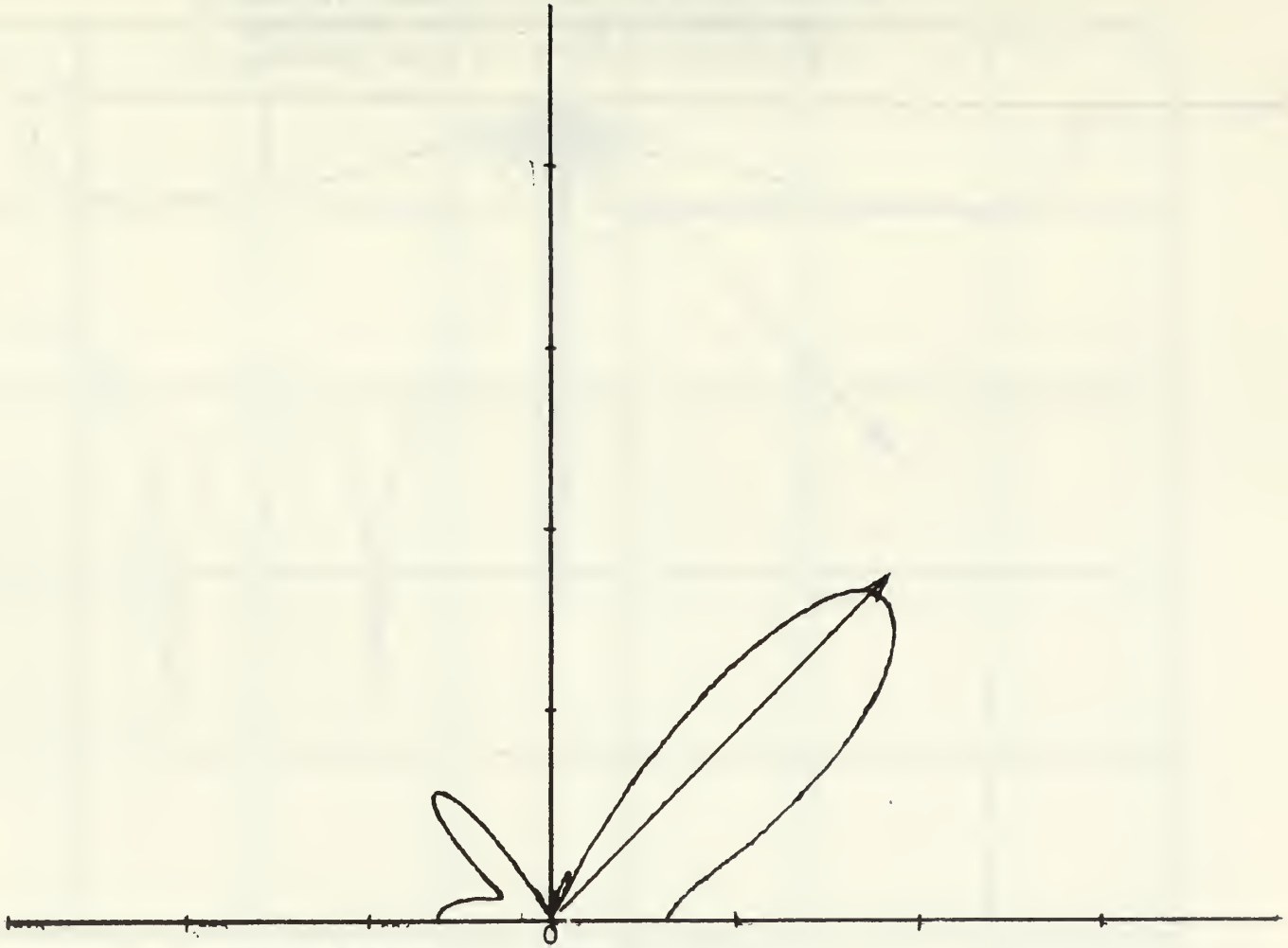
Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 45^{\circ}$ after adapting
for 15 cycles of the desired frequency 500 Hz.



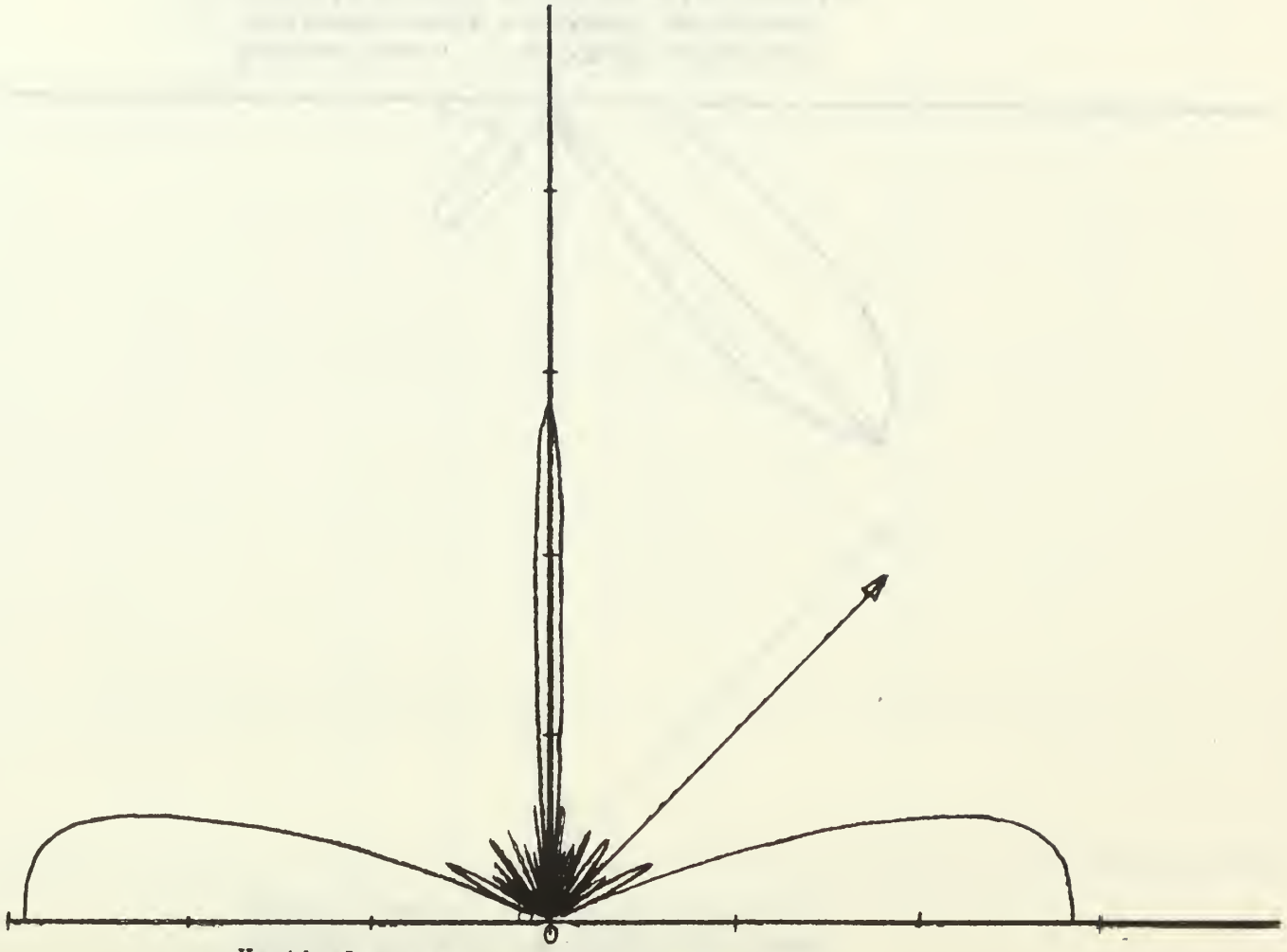
Vertical scale =4.00E-01 units/inch
Horizontal scale =4.00E-01 units/inch
Directivity pattern at 100 Hz after adapting
for 300 cycles of the desired frequency 500 Hz.



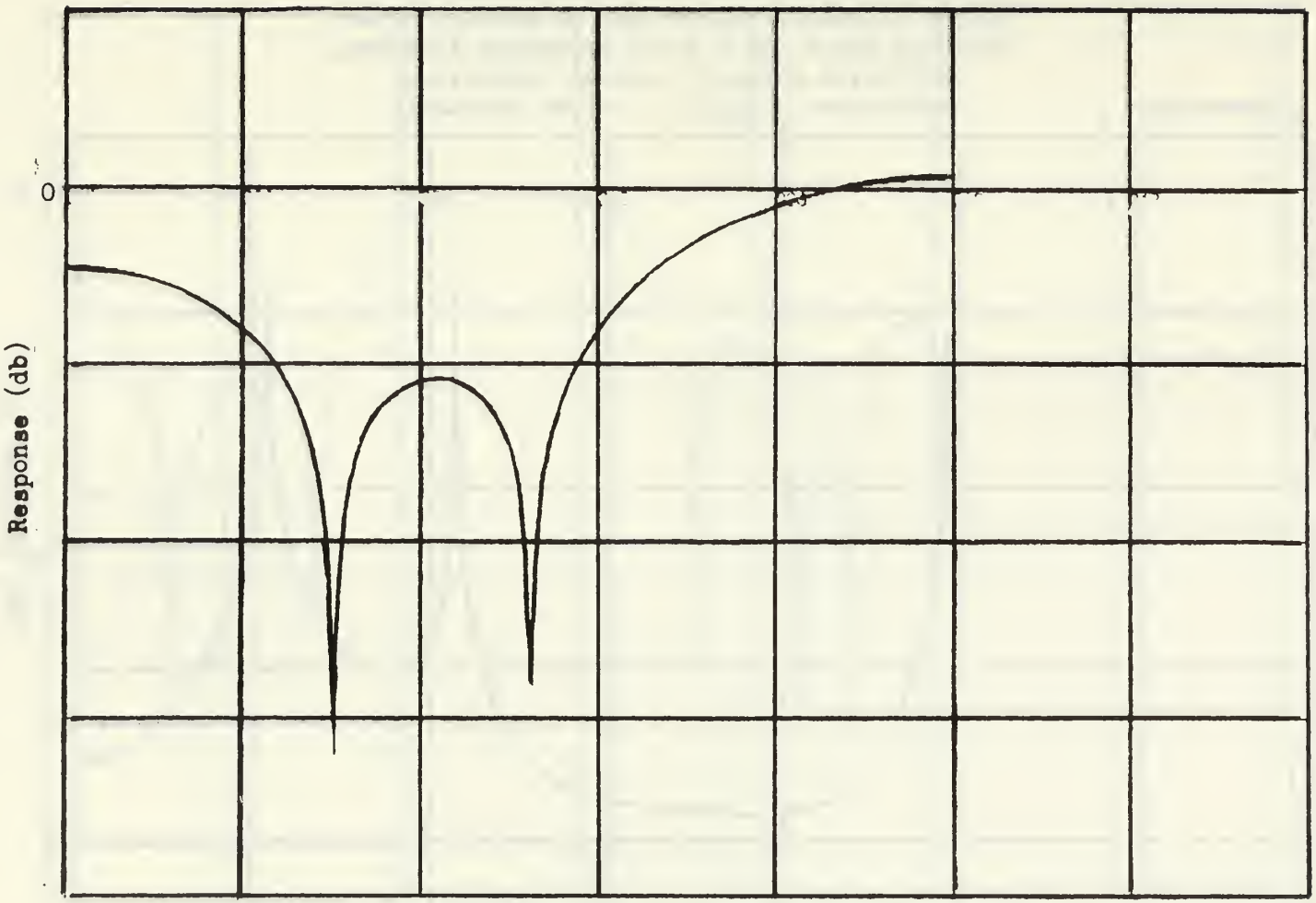
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 200 Hz after adapting
for 300 cycles of the desired frequency 500 Hz.



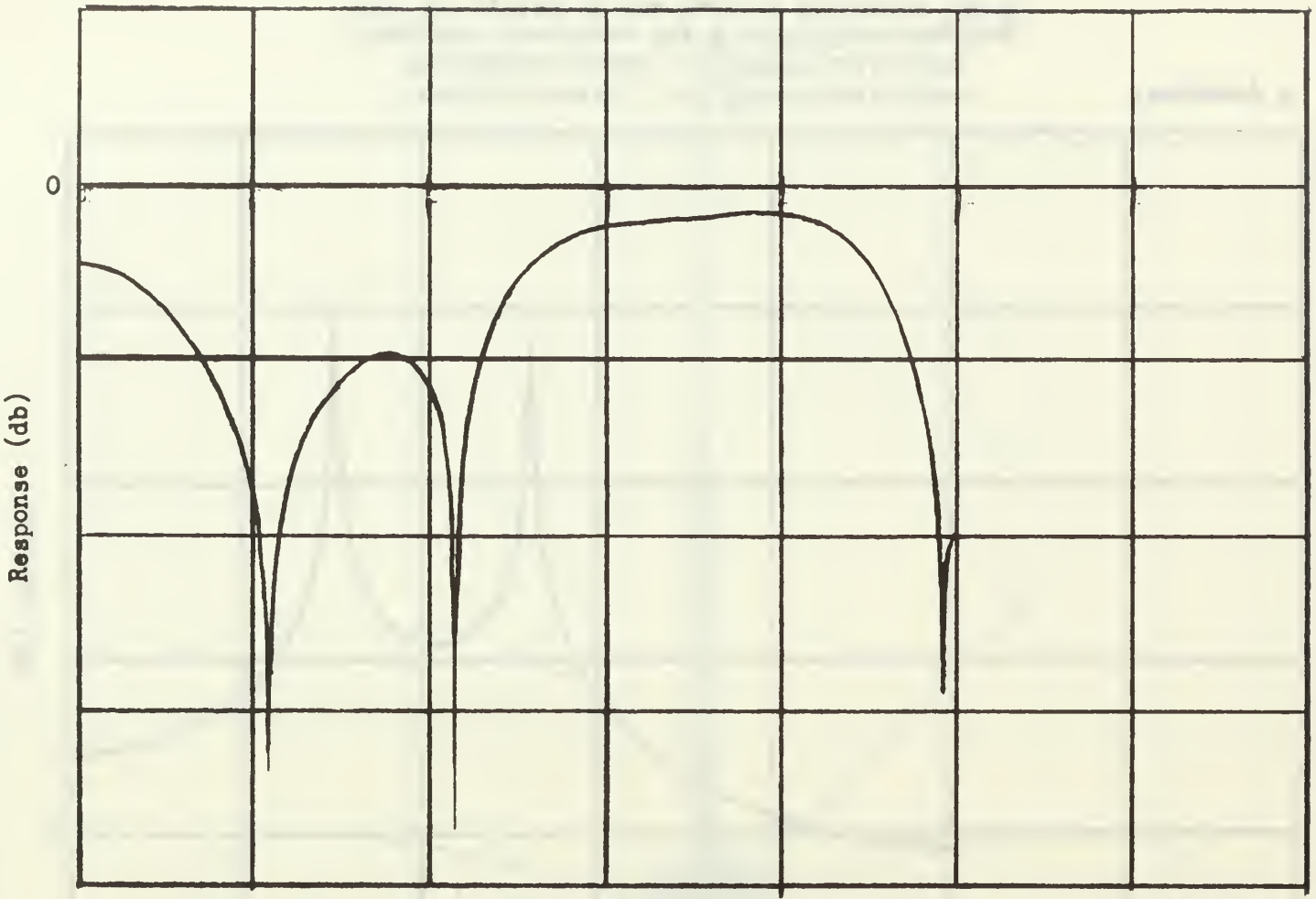
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 500 Hz after adapting
for 300 cycles of the desired frequency 500 Hz.



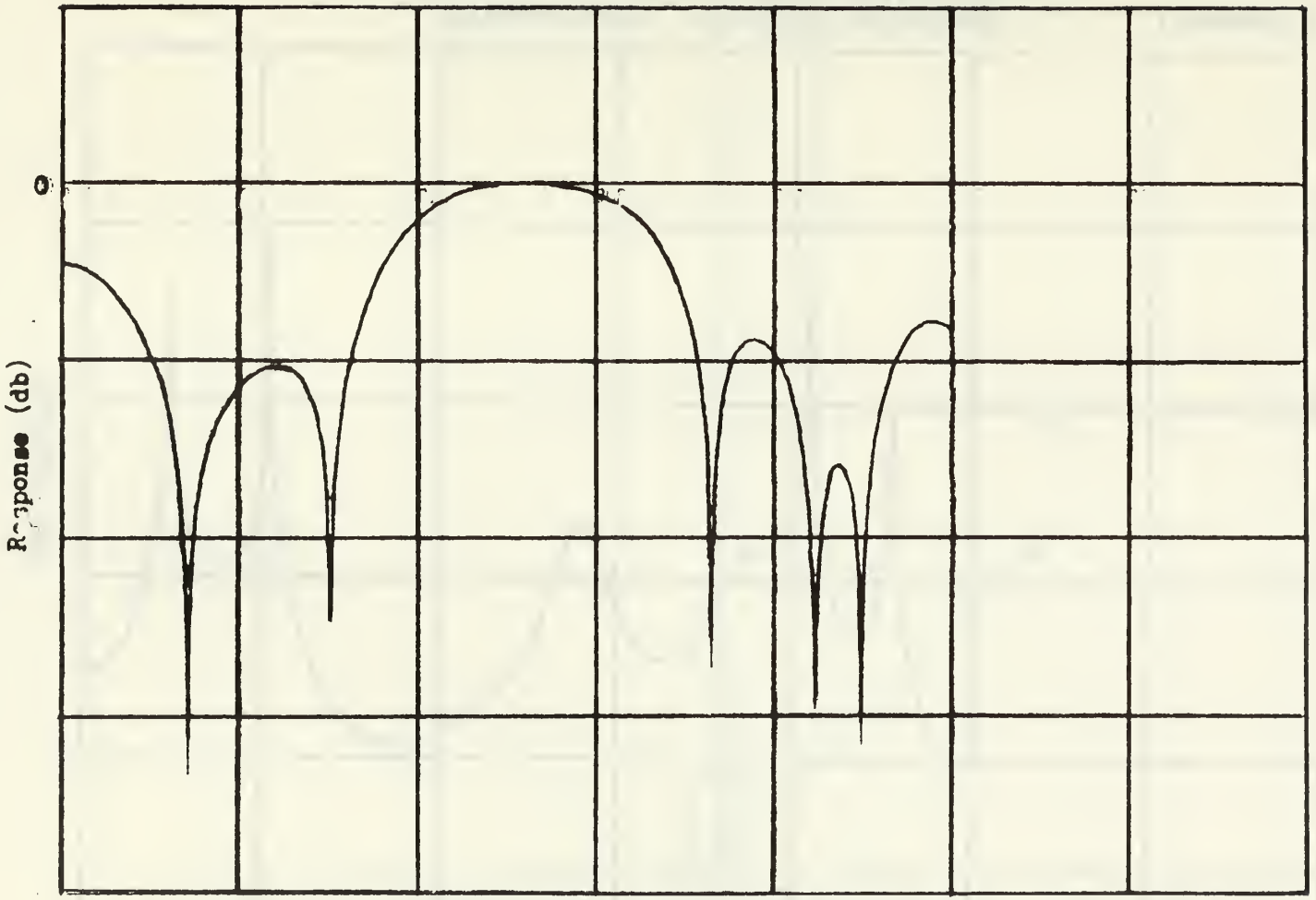
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 1000 Hz after adapting
for 300 cycles of the desired frequency 500 Hz.



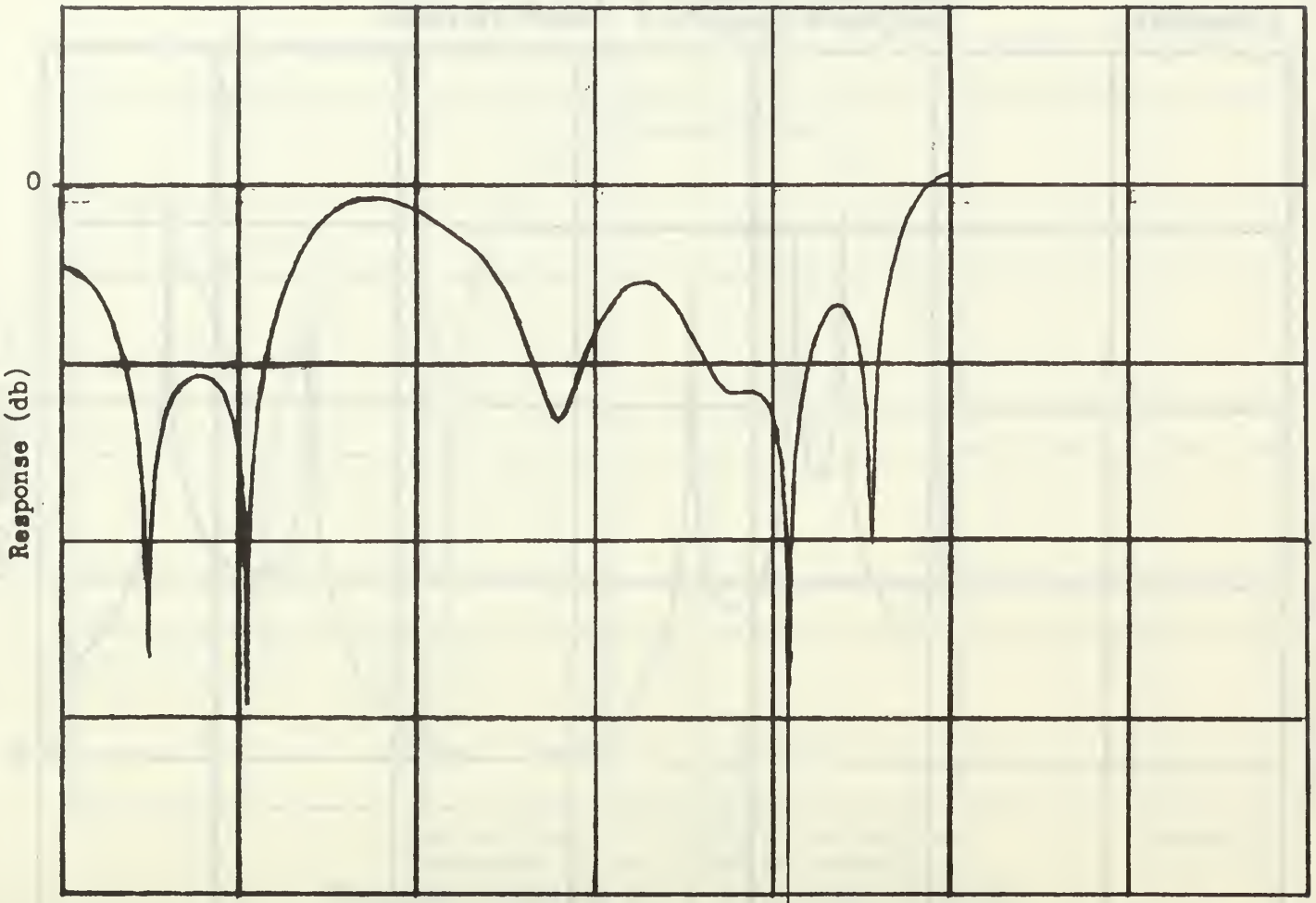
Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 0^{o}$ after adapting
for 300 cycles of the desired frequency 500 Hz.



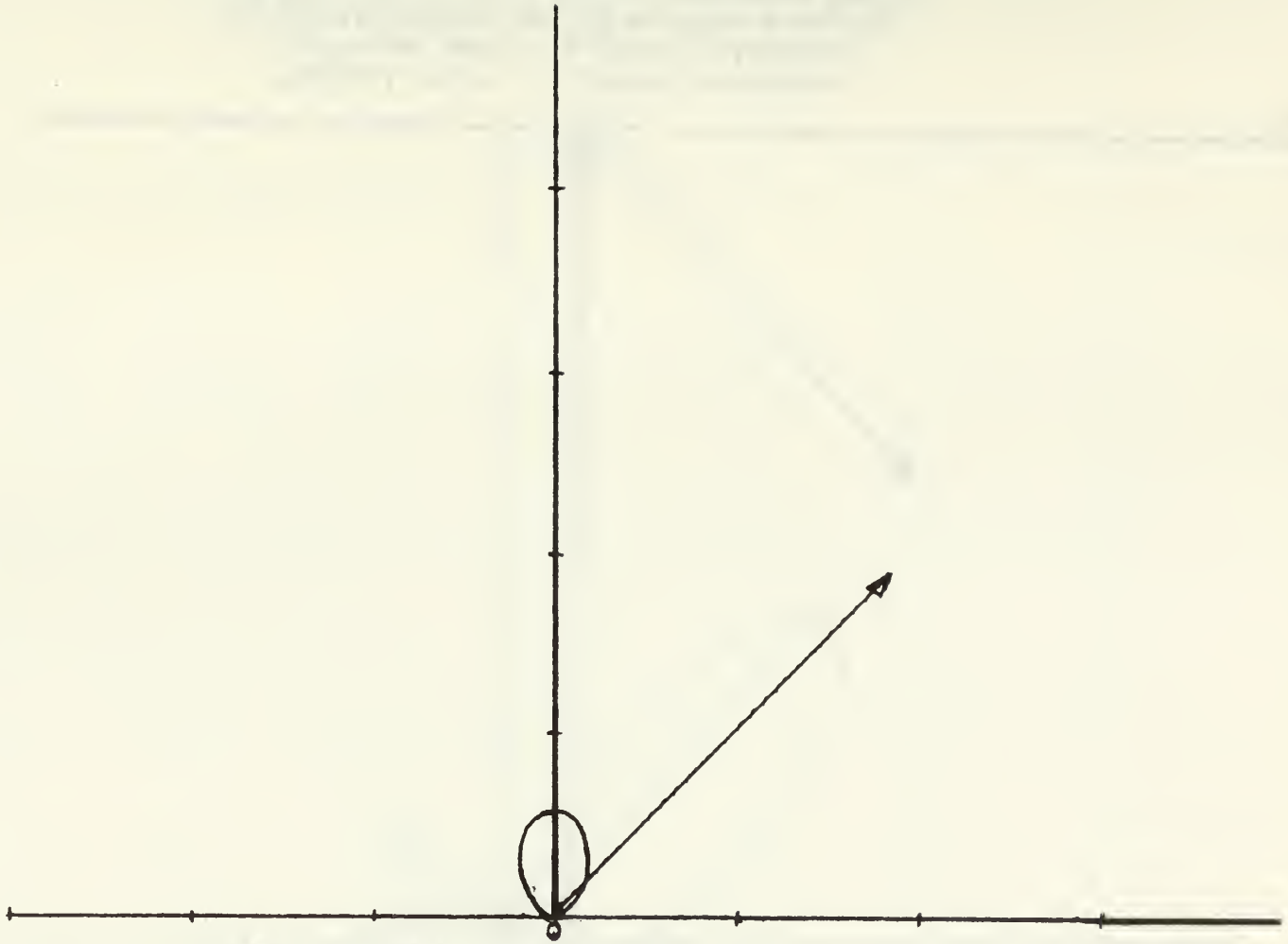
Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 30^{o}$ after adapting
for 300 cycles of the desired frequency 500 Hz.



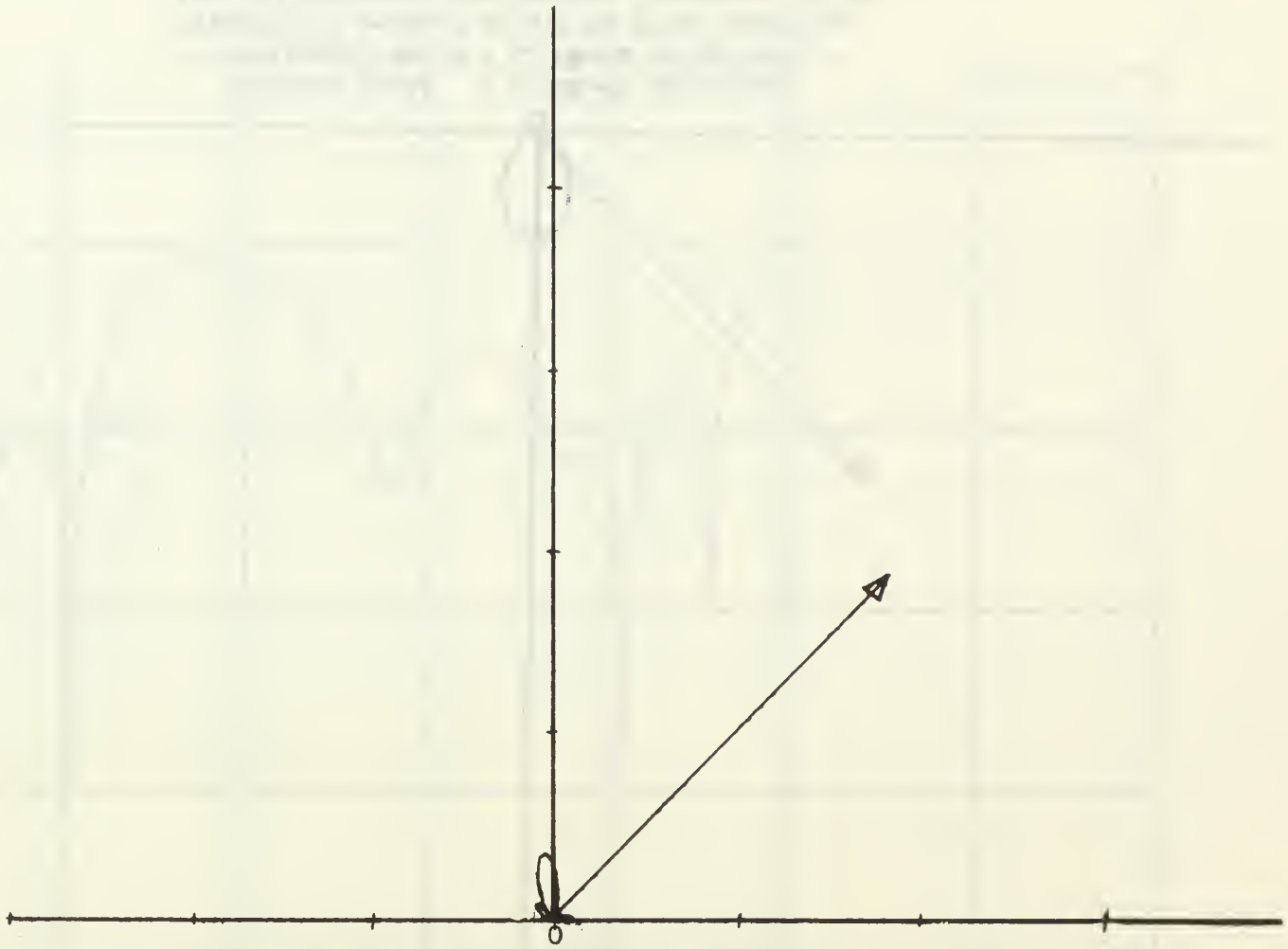
Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 45^{\circ}$ after adapting
for 300 cycles of the desired frequency 500 Hz.



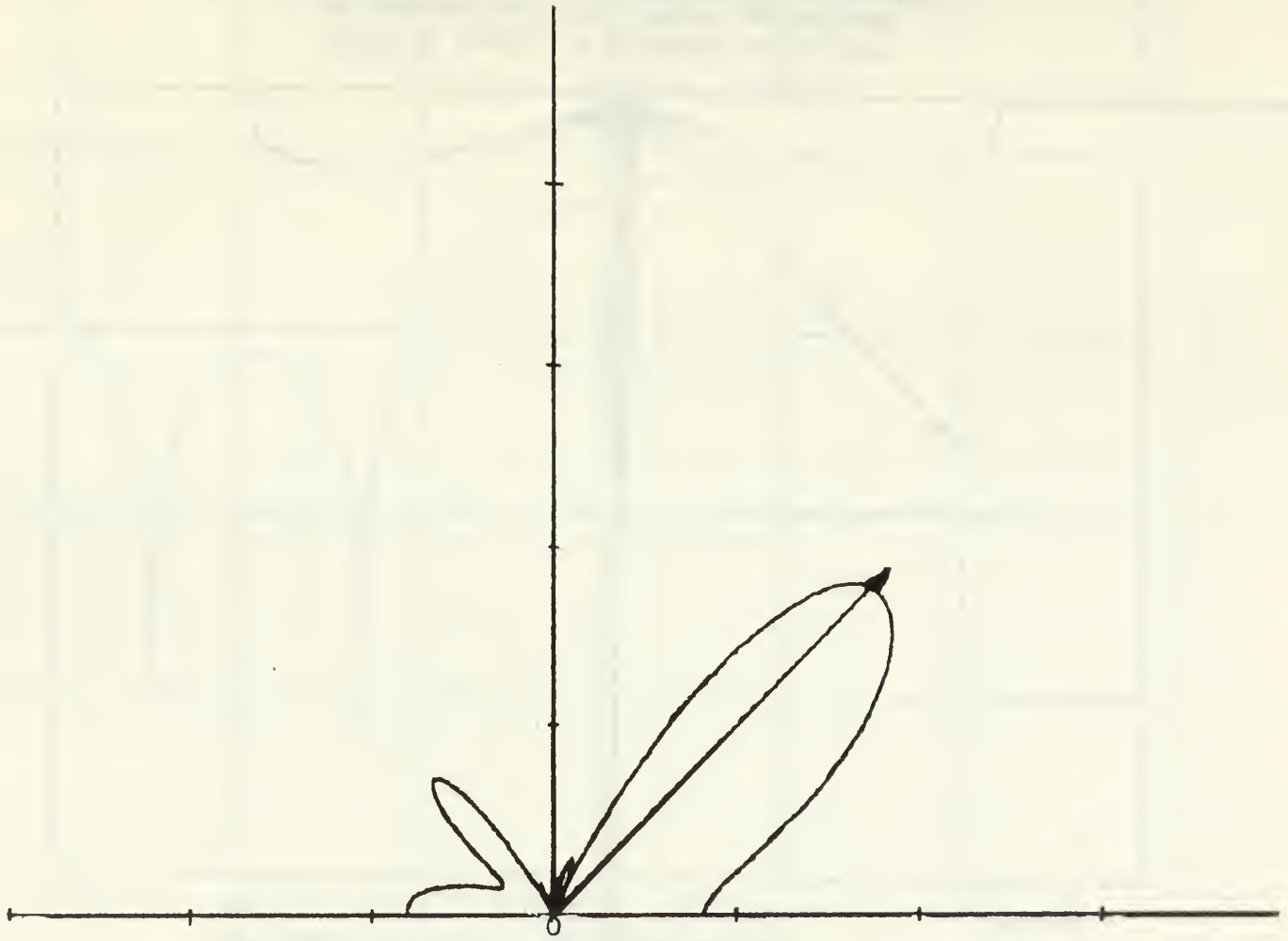
Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 88.8^{o}$ after adapting for 300 cycles of the desired frequency 500 Hz.



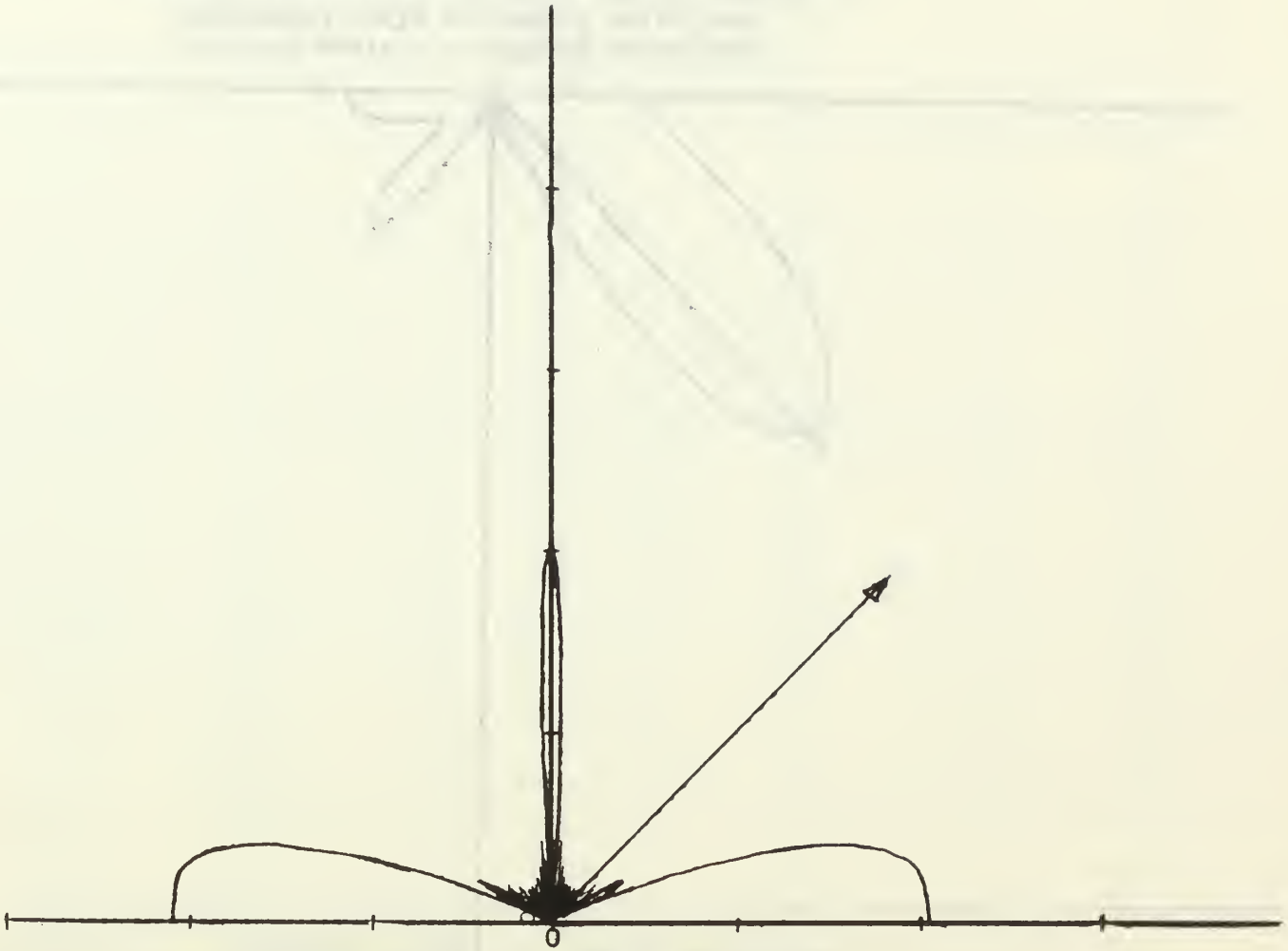
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 100 Hz after adapting
for 600 cycles of the desired frequency 500 Hz.



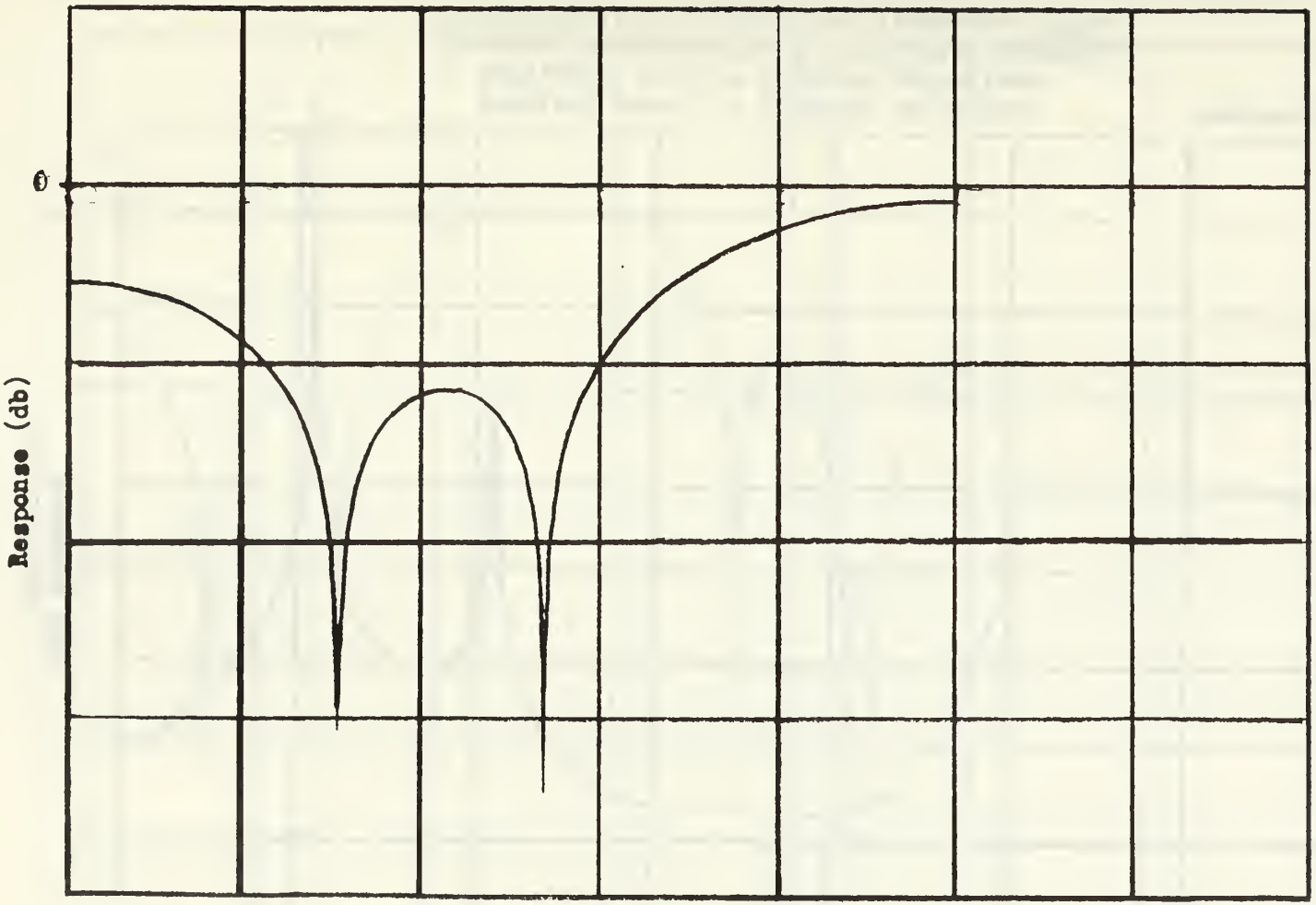
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 200 Hz after adapting
for 600 cycles of the desired frequency 500 Hz.



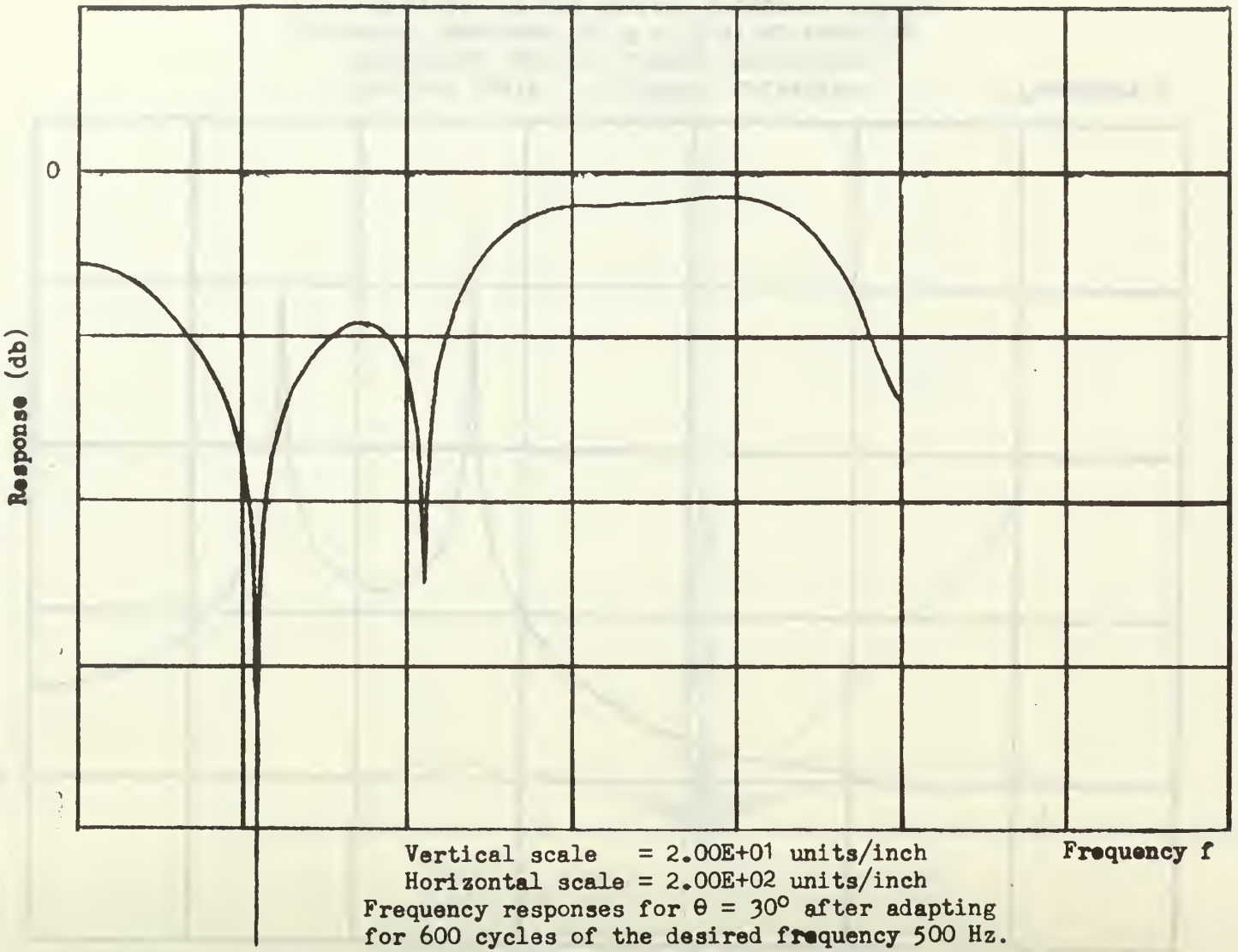
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 500 Hz after adapting
for 600 cycles of the desired frequency 500 Hz.



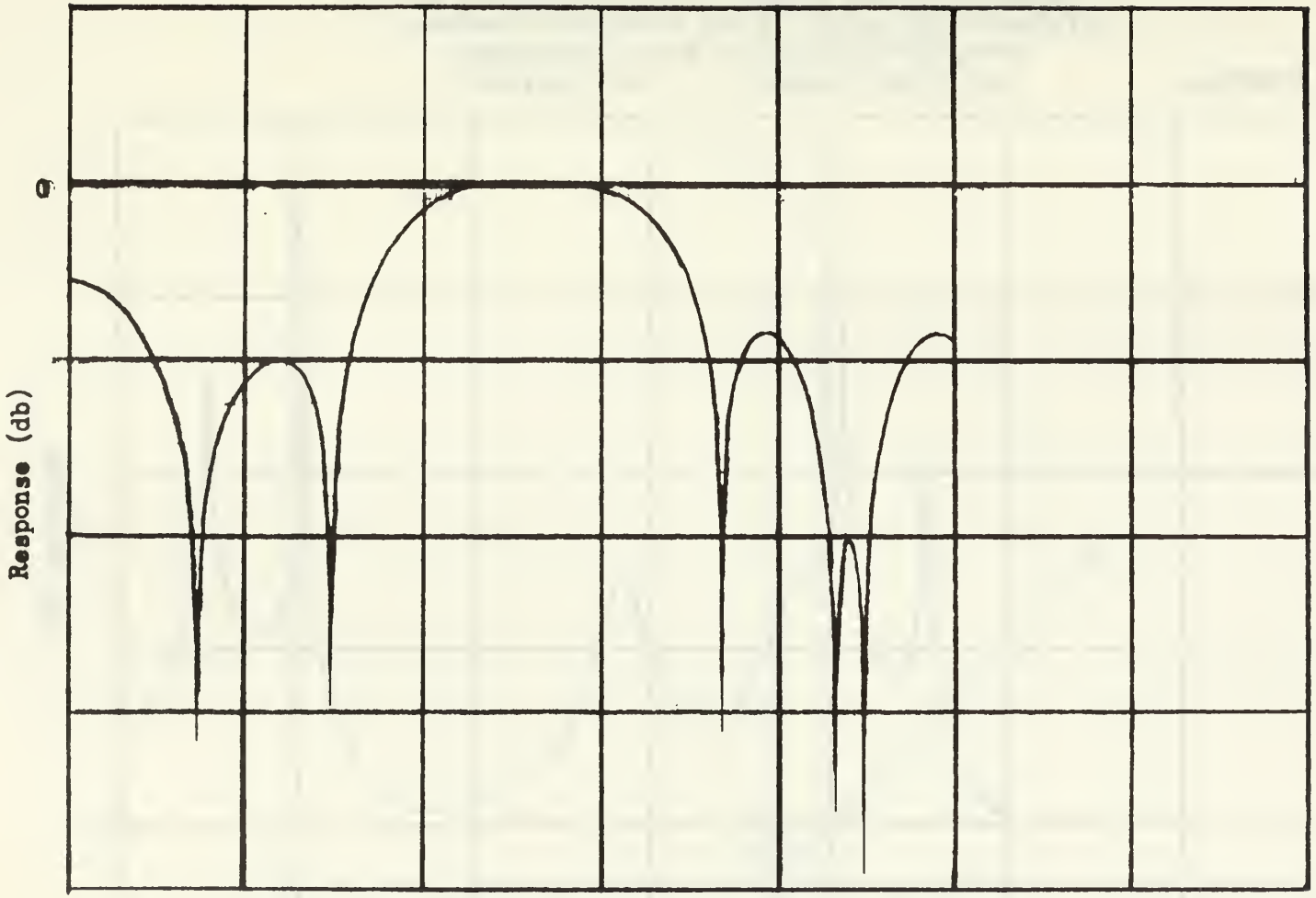
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 1000 Hz after adapting
for 600 cycles of the desired frequency 500 Hz.



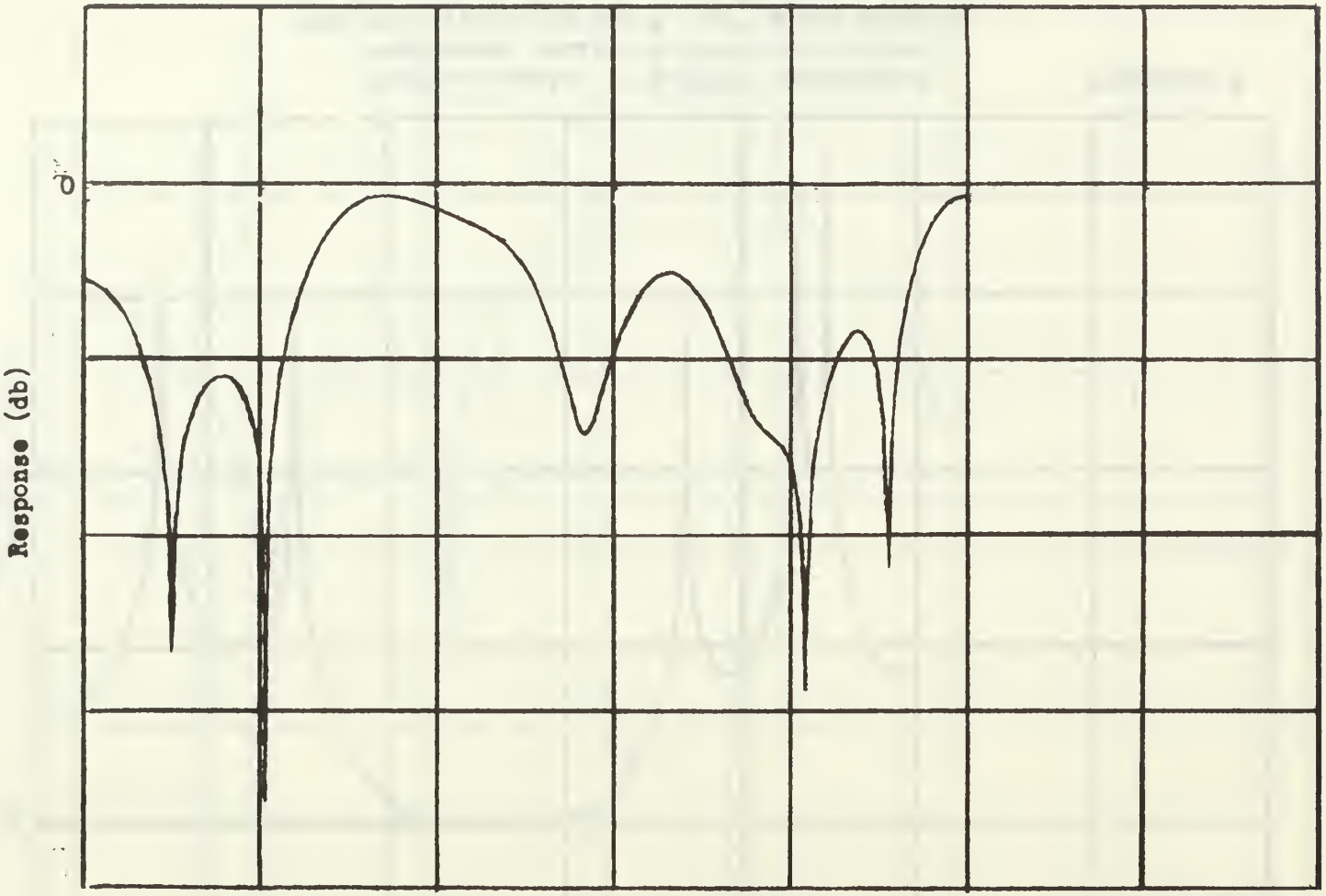
Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 0^{\circ}$ after adapting
for 600 cycles of the desired frequency 500 Hz.



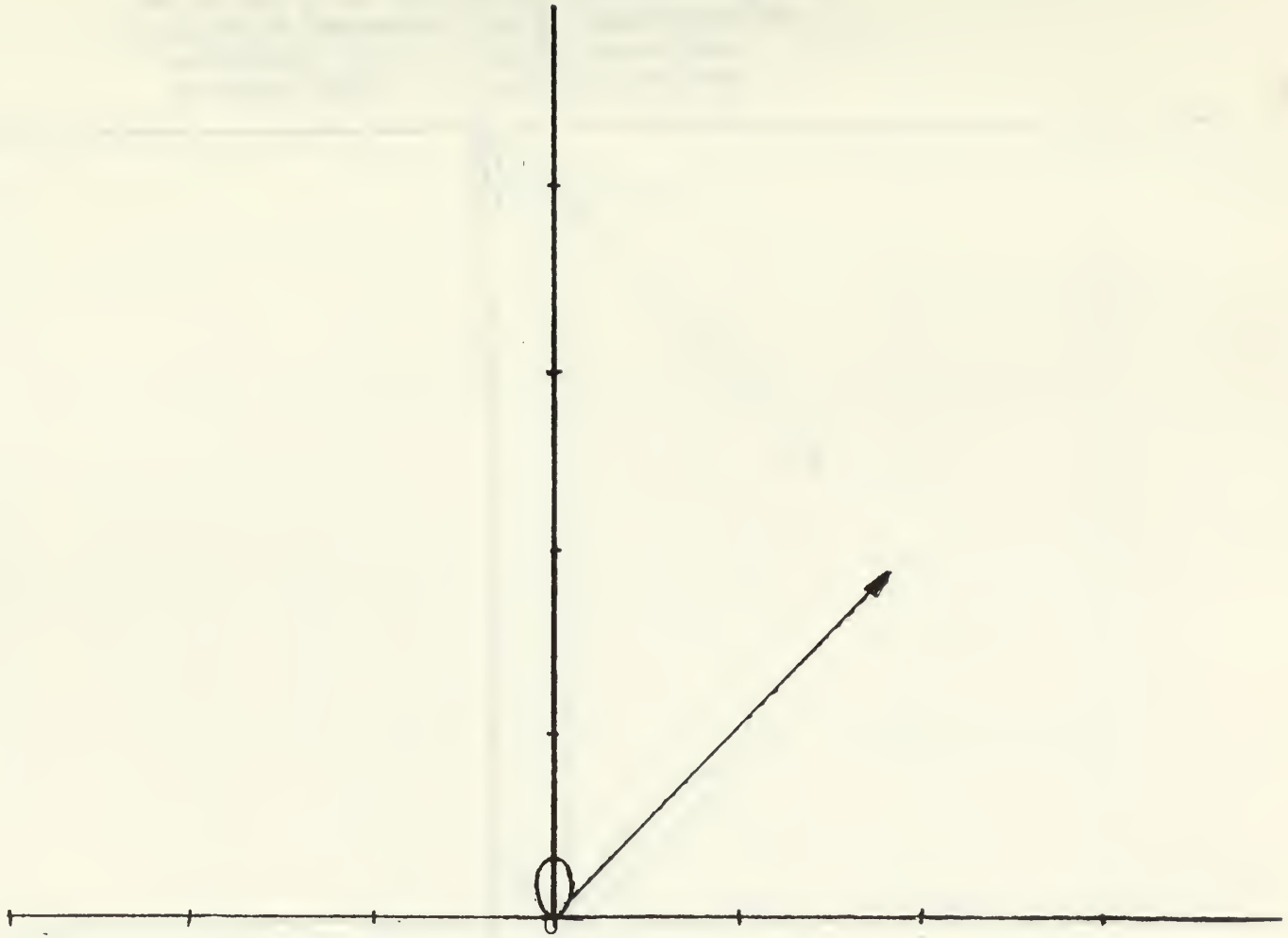
Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 30^{\circ}$ after adapting for 600 cycles of the desired frequency 500 Hz.



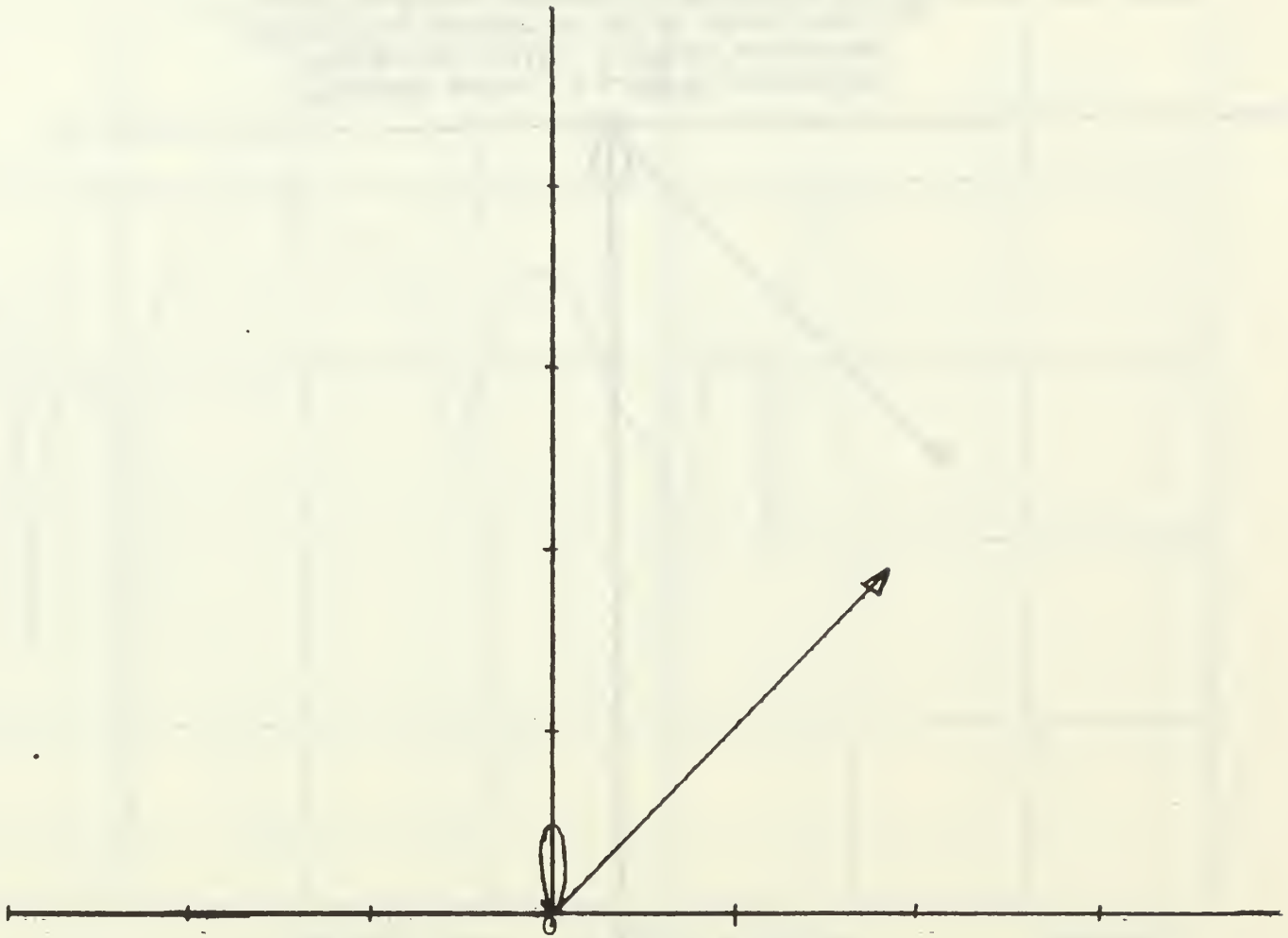
Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 45^{\circ}$ after adapting
for 600 cycles of the desired frequency 500 Hz.



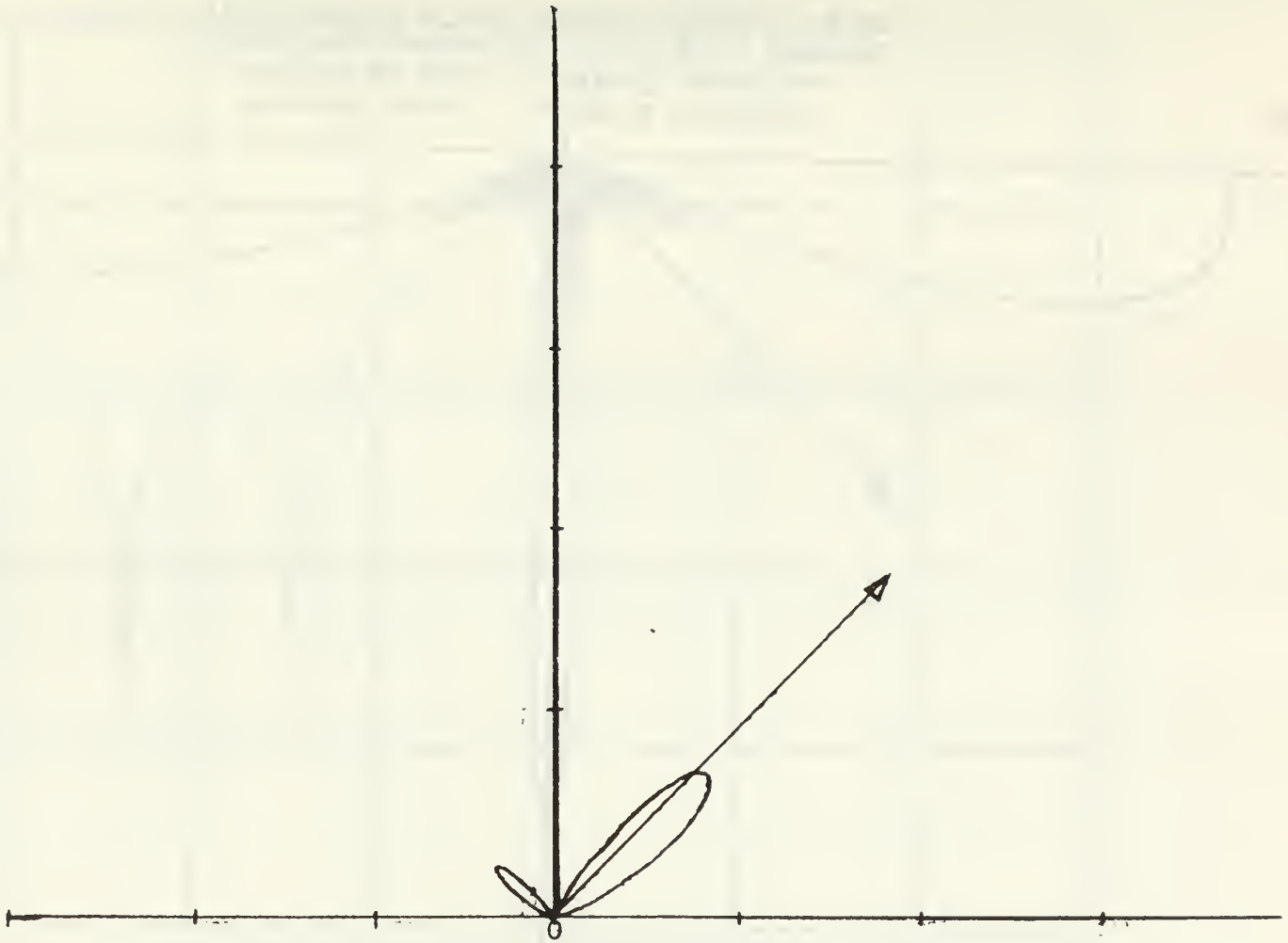
Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 88.8^{\circ}$ after adapting for 600 cycles of the desired frequency 500 Hz.



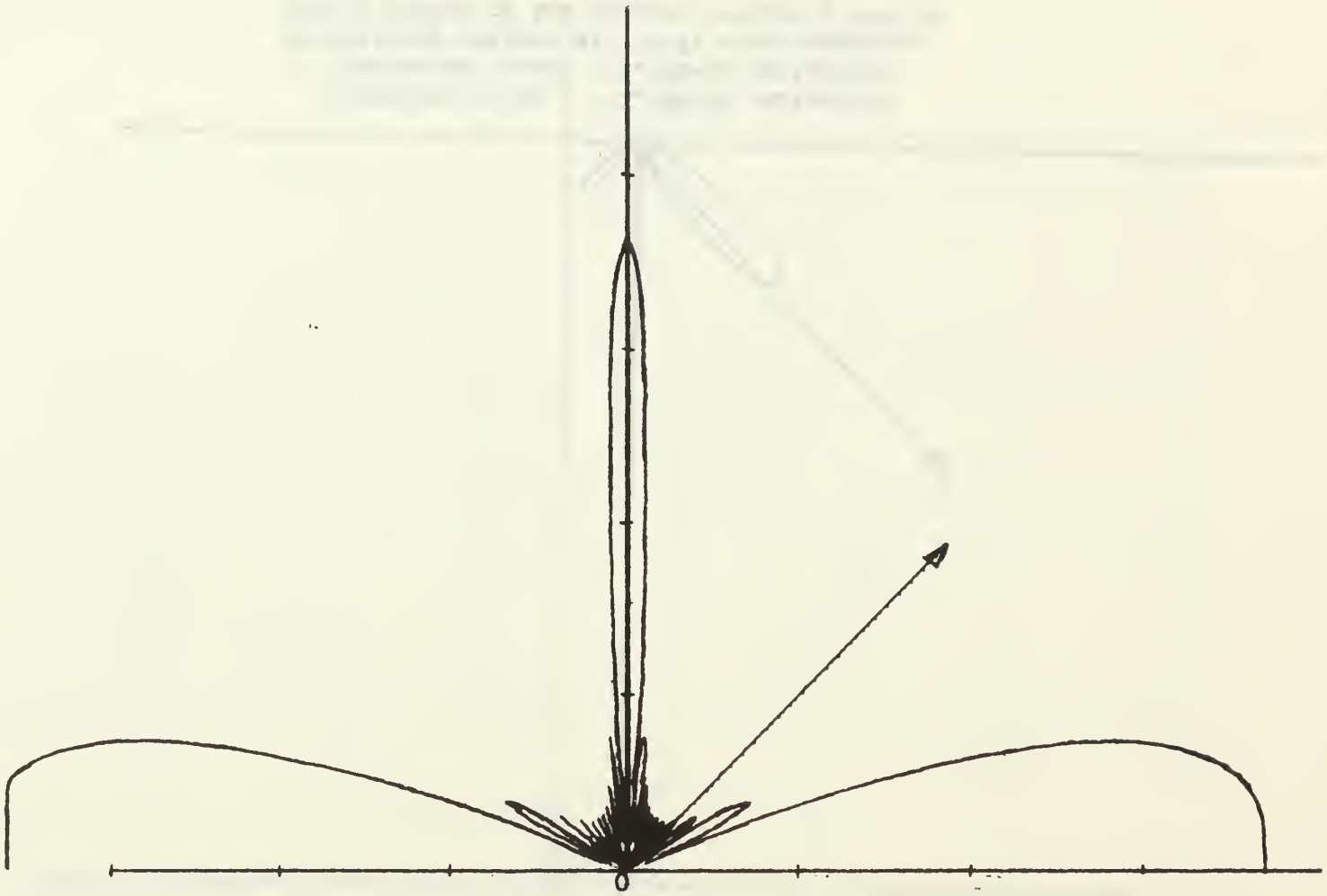
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 100 Hz after adapting
for 15 cycles of the desired frequency 500 Hz.



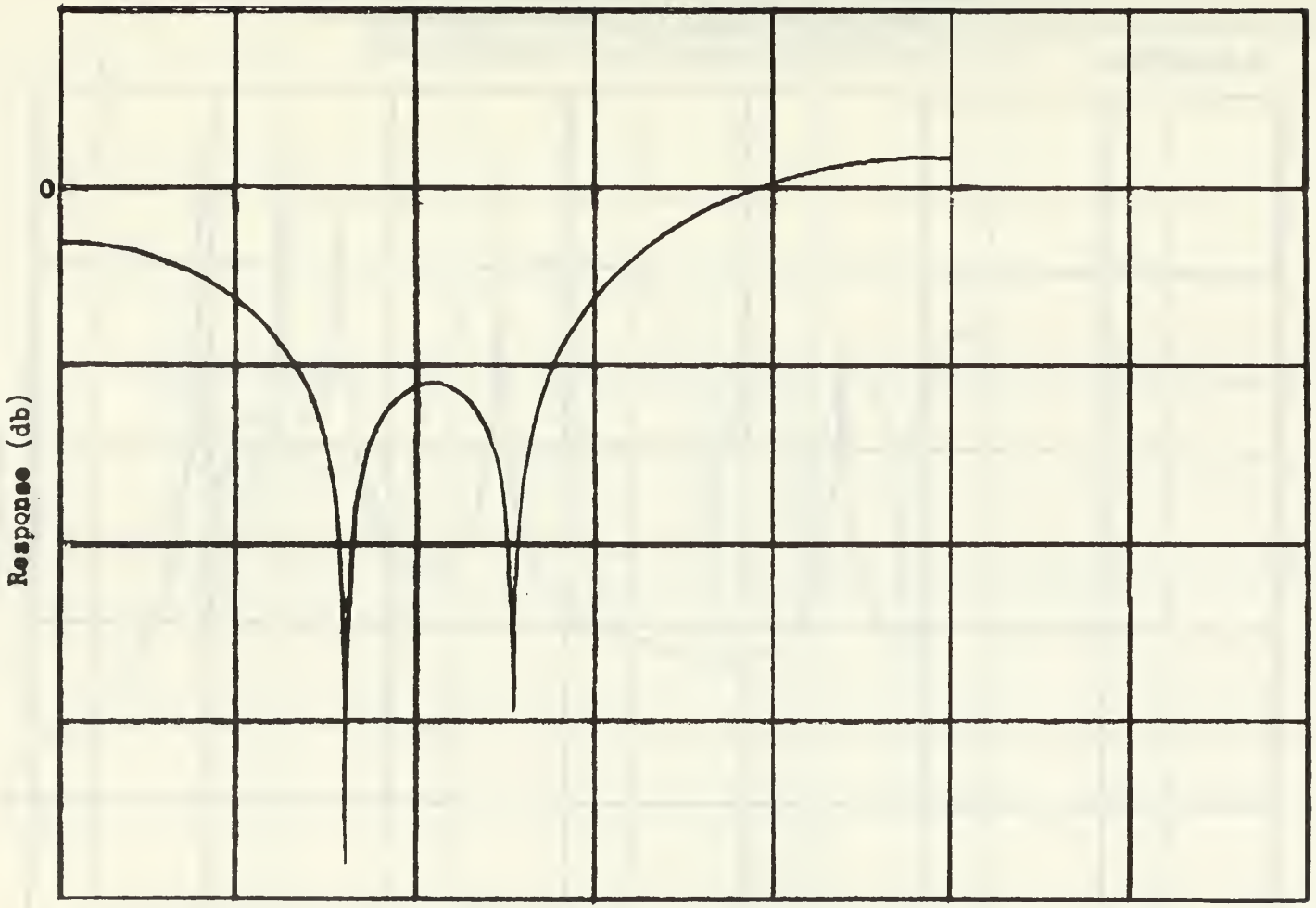
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 200 Hz after adapting
for 15 cycles of the desired frequency 500 Hz.



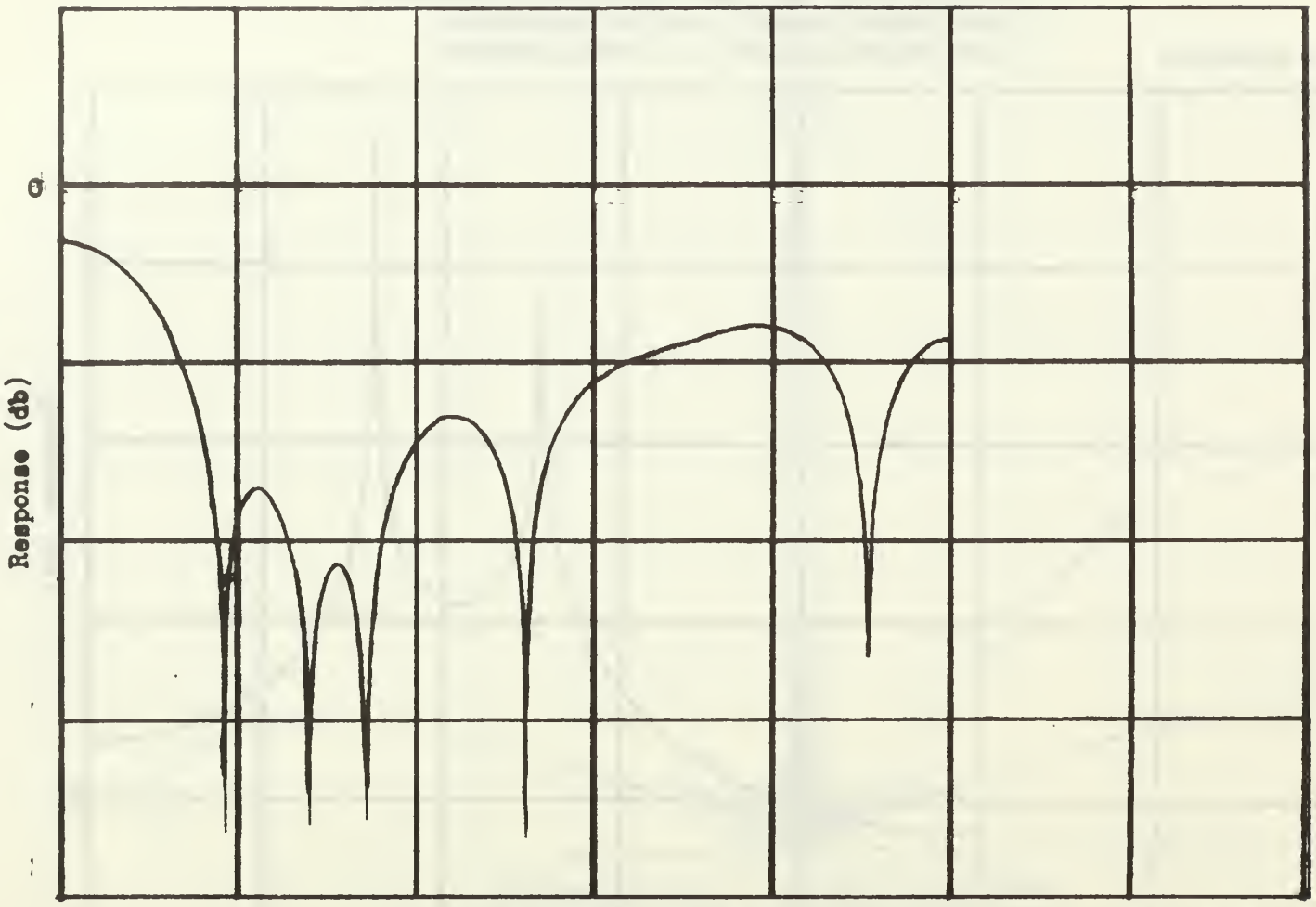
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 500 Hz after adapting
for 15 cycles of the desired frequency 500 Hz.



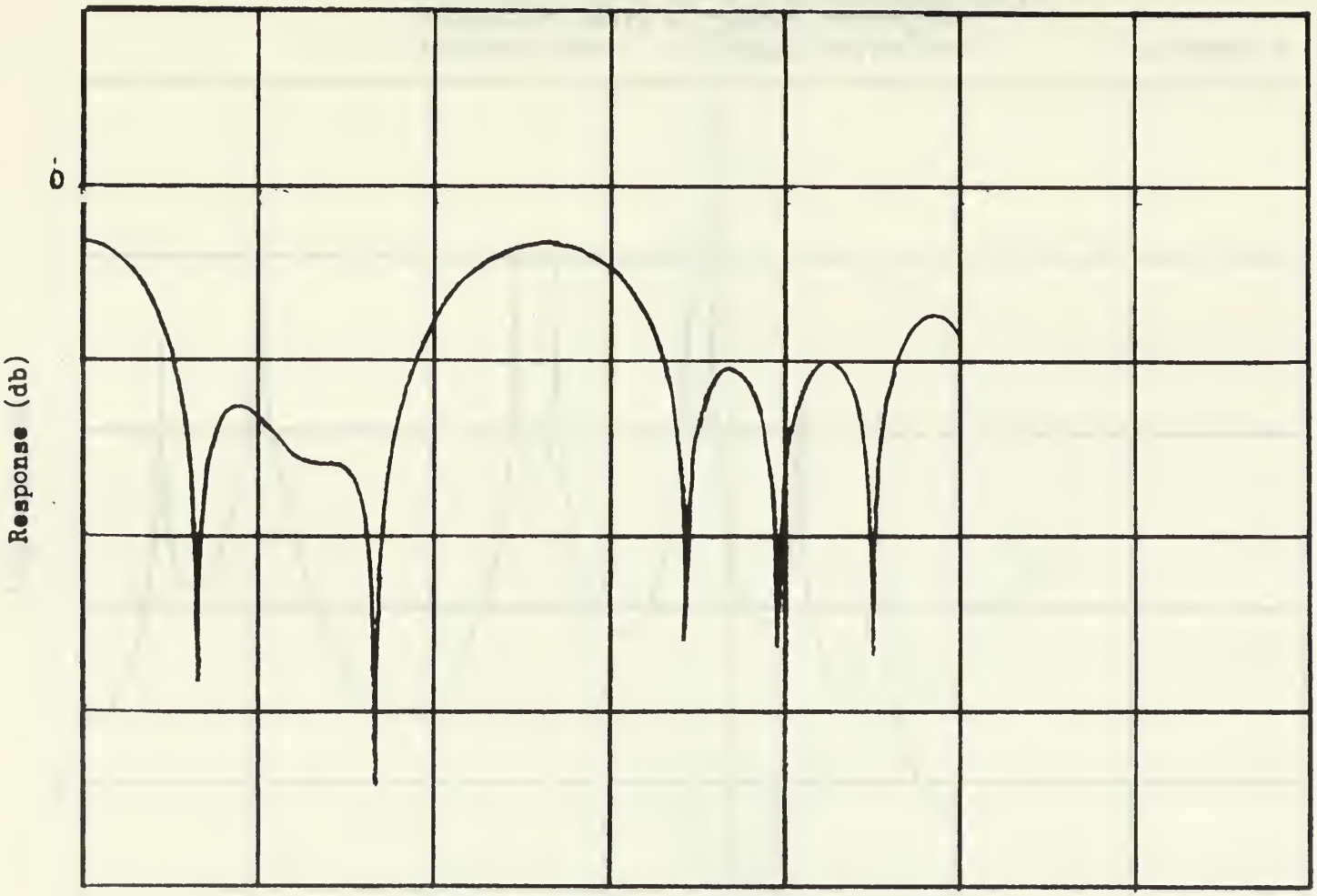
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 1000 Hz after adapting
for 15 cycles of the desired frequency 500 Hz.



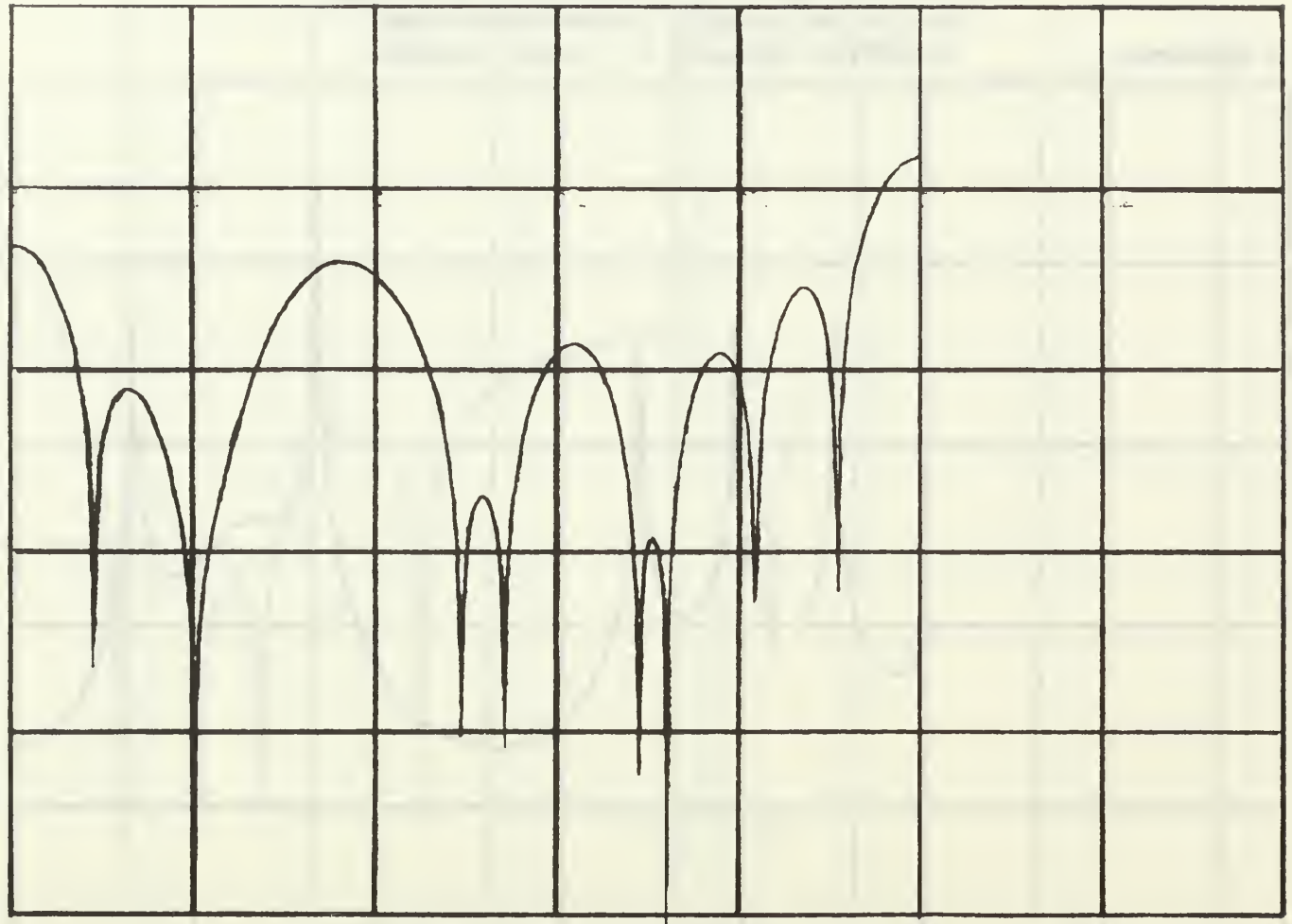
Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 0^{\circ}$ after adapting
for 15 cycles of the desired frequency 500 Hz.



Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 30^{\circ}$ after adapting for 15 cycles of the desired frequency 500 Hz.

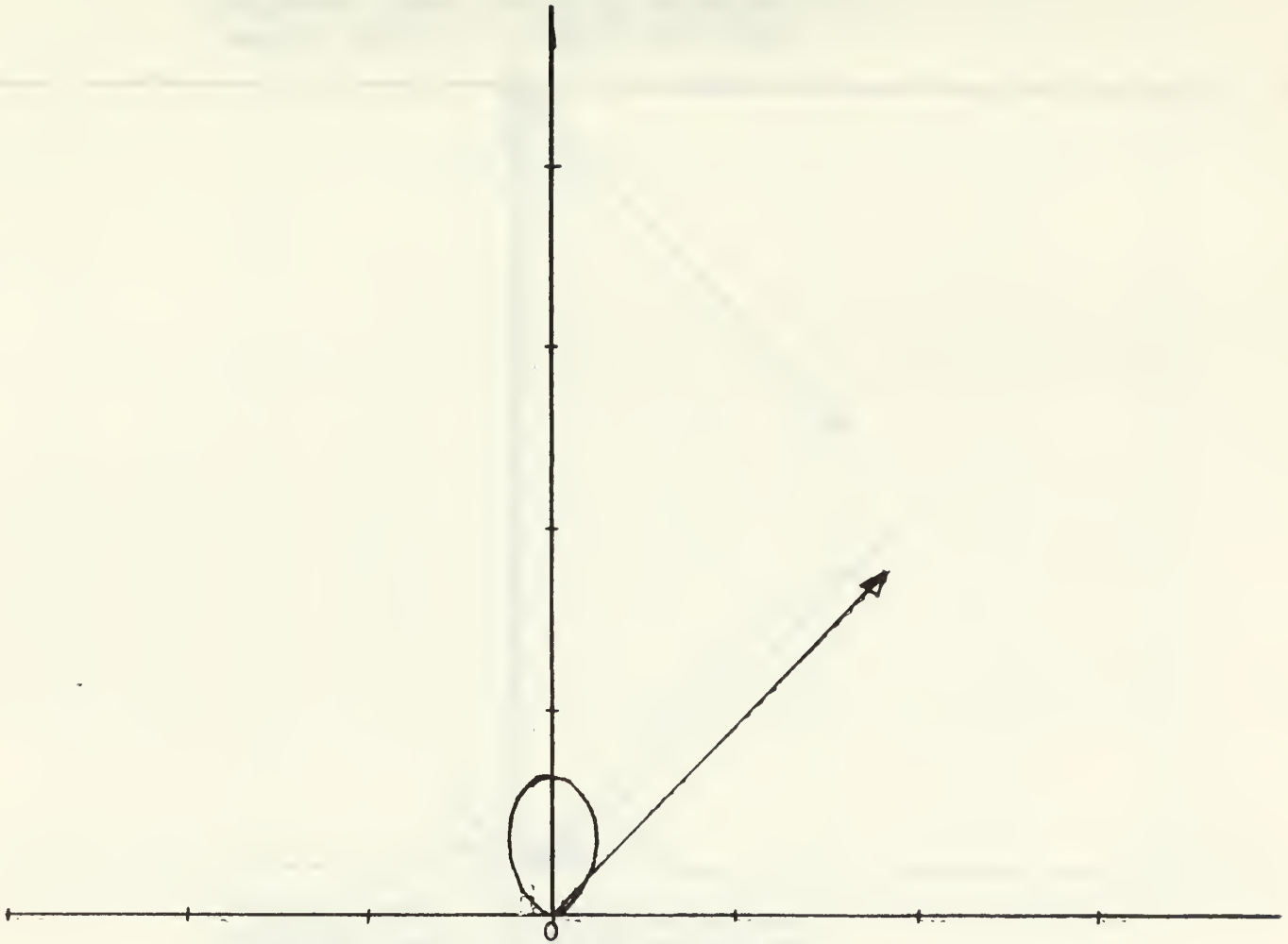


Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 45^{\circ}$ after adapting for 15 cycles of the desired frequency 500 Hz.

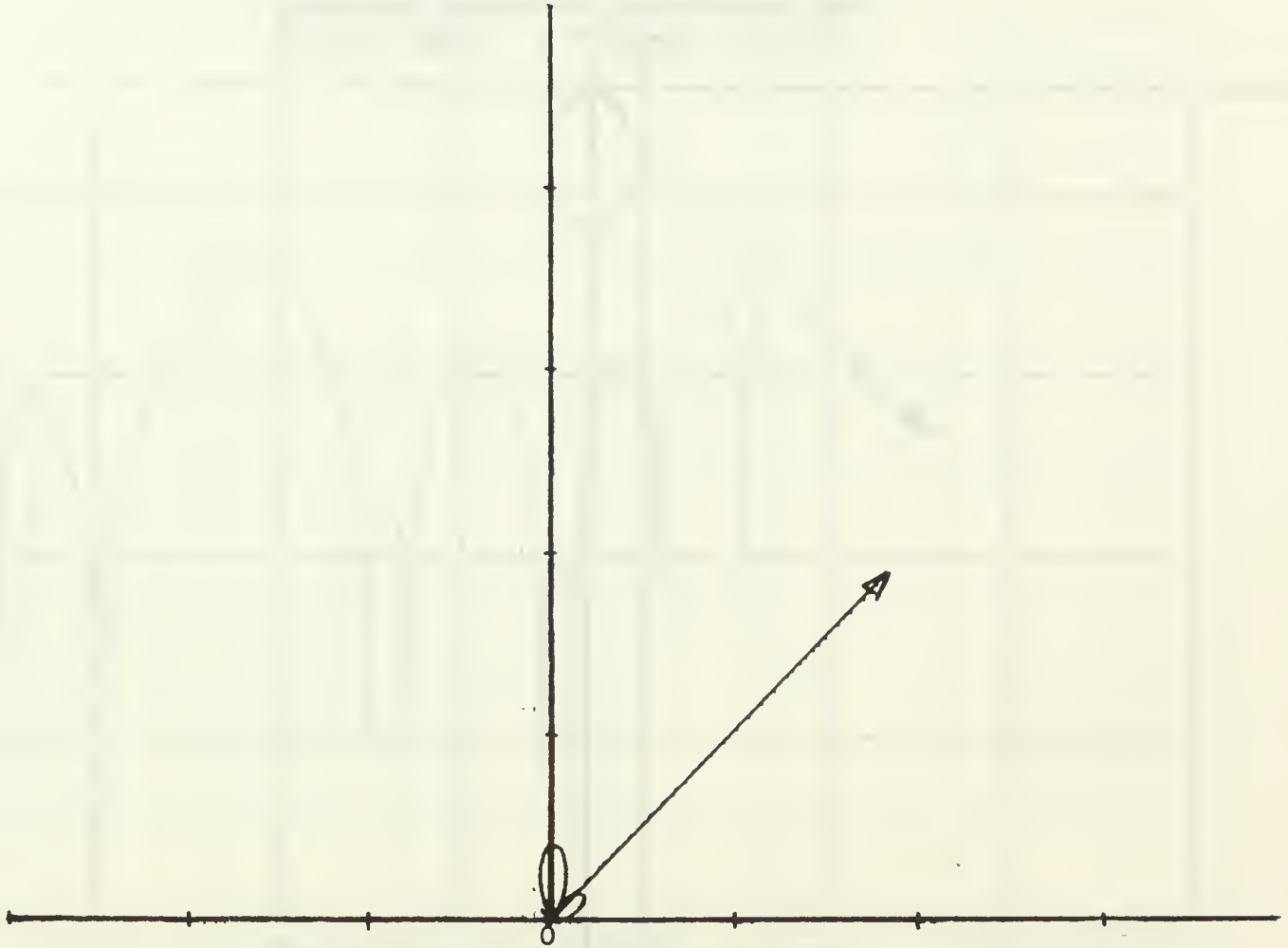


Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch

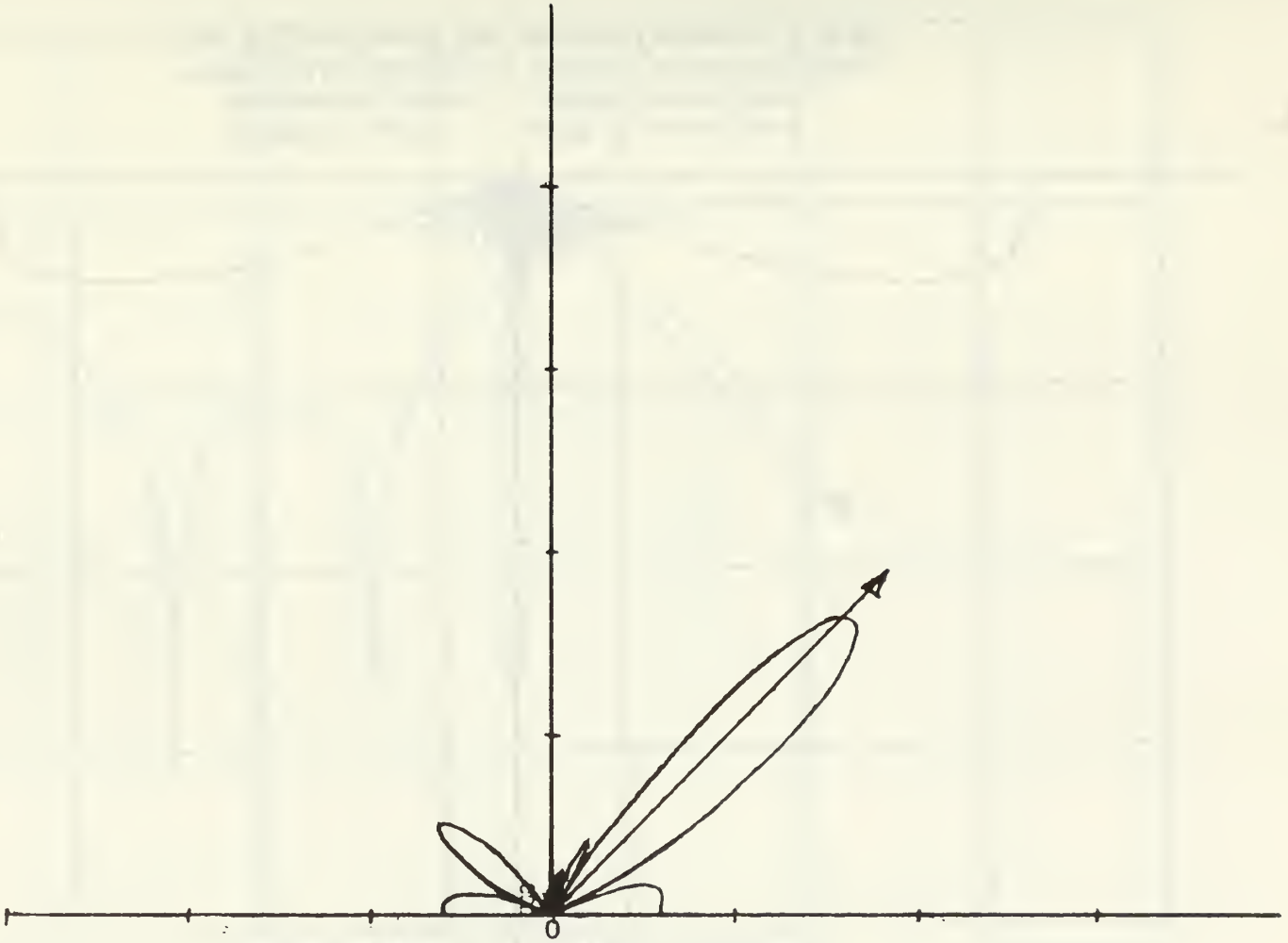
Frequency response for $\theta = 88.8^{\circ}$ after adapting for 15 cycles of the desired frequency 500 Hz.



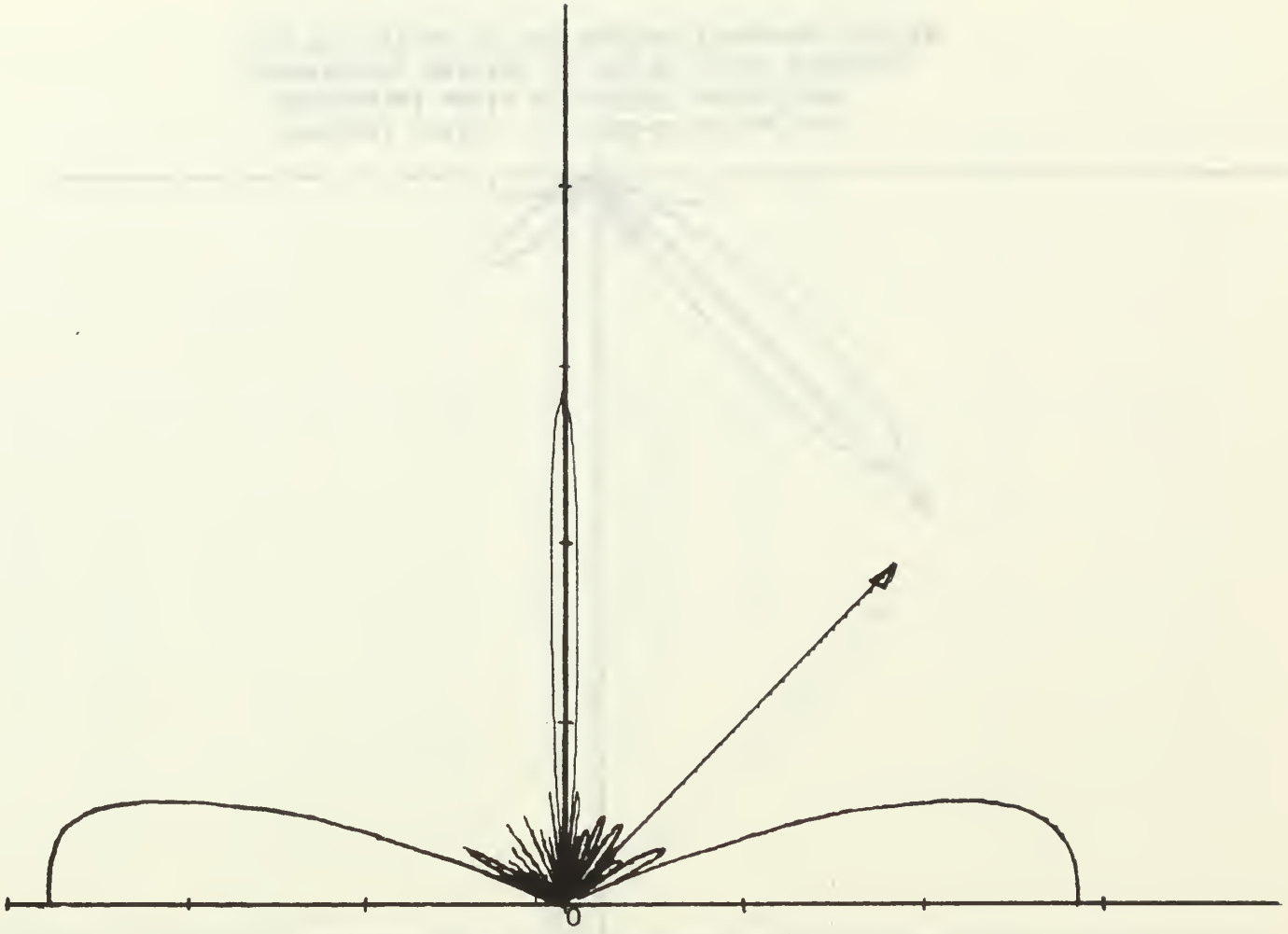
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 100 Hz after adapting
for 300 cycles of the desired frequency 500 Hz.



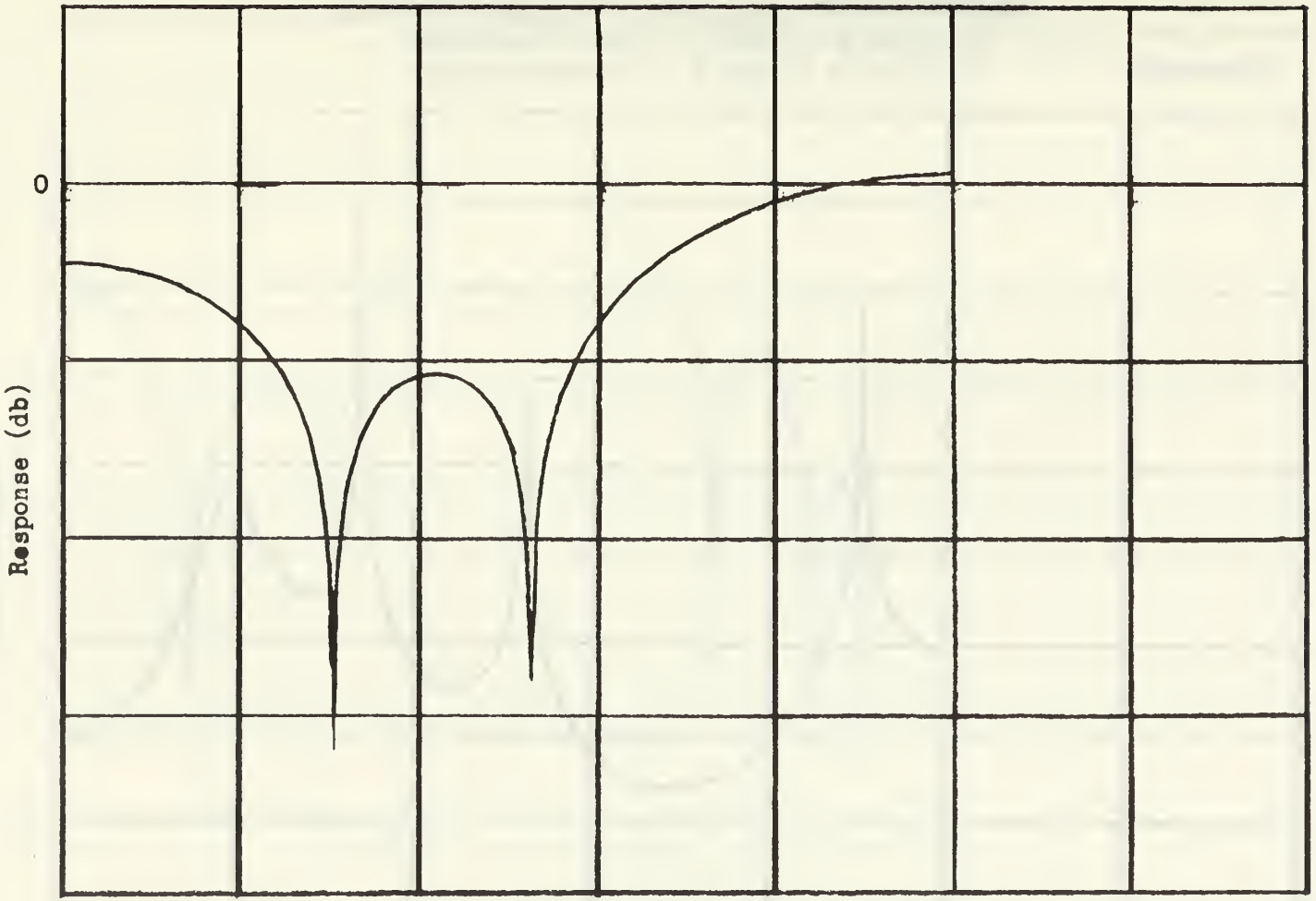
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 200 Hz after adapting
for 300 cycles of the desired frequency 500 Hz.



Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 500 Hz after adapting
for 300 cycles of the desired frequency 500 Hz.

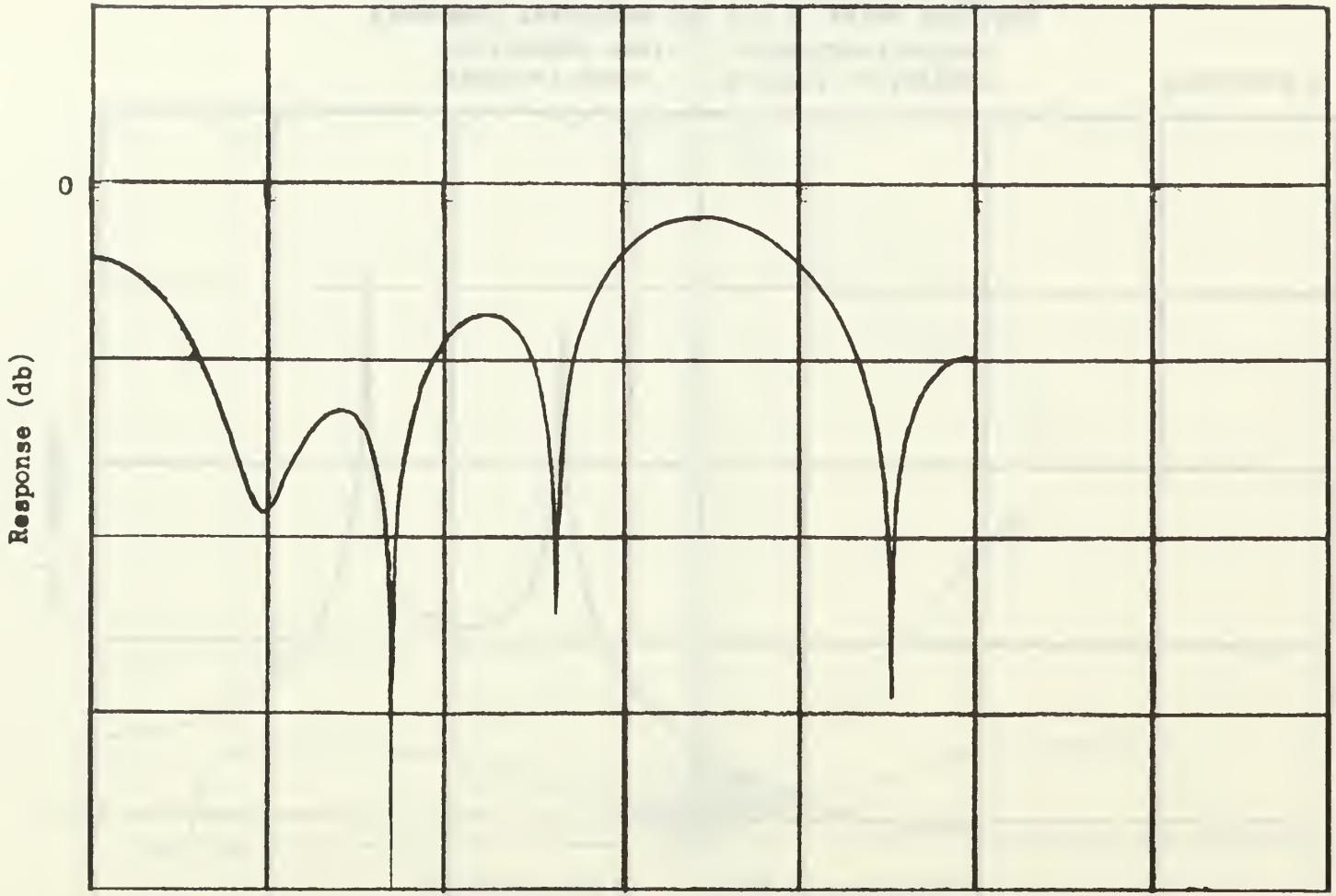


Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 1000 Hz after adapting
for 300 cycles of the desired frequency 500 Hz.

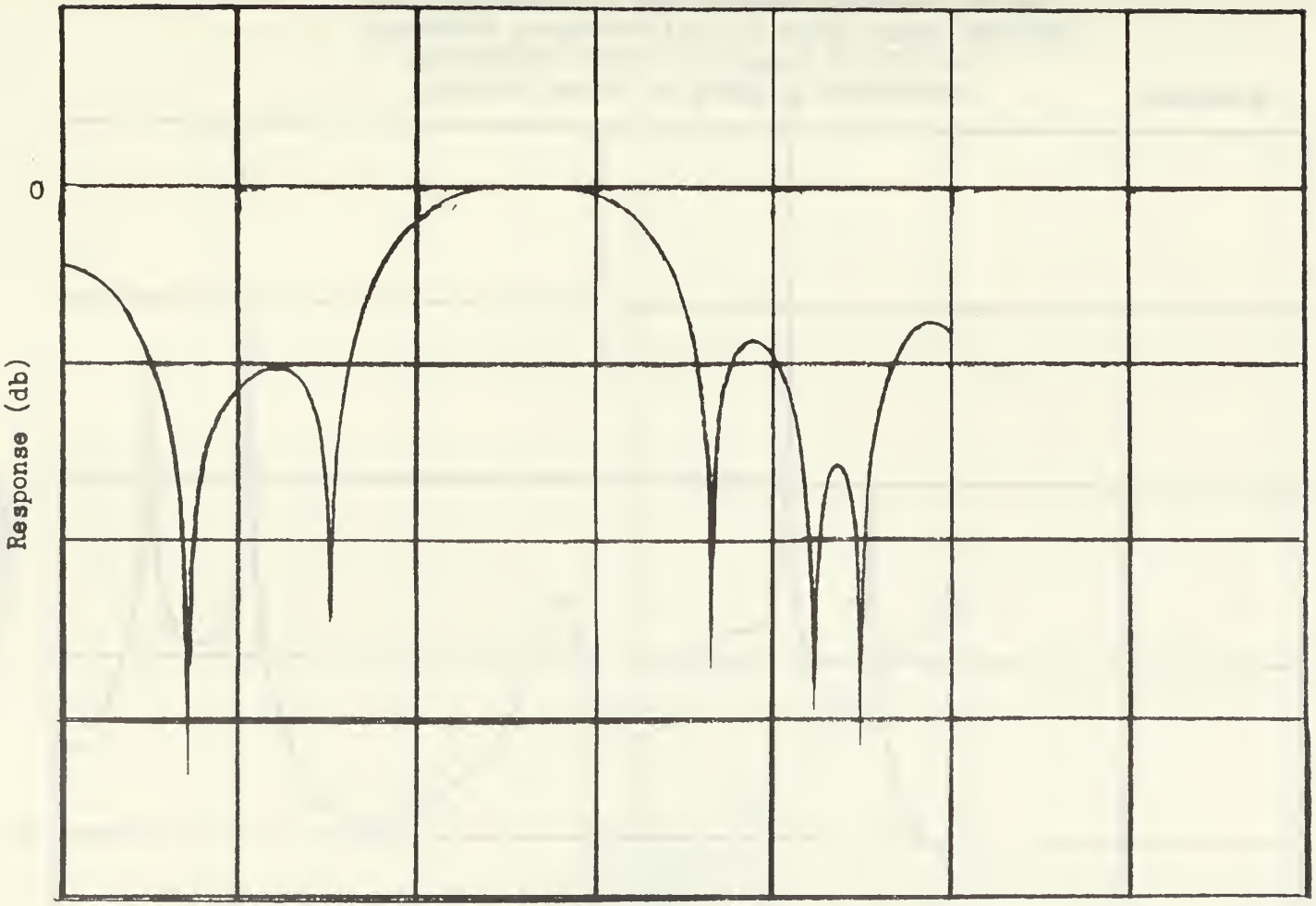


Vertical scale = 2.00E01 units/inch
Horizontal scale = 2.00E+02units/inch

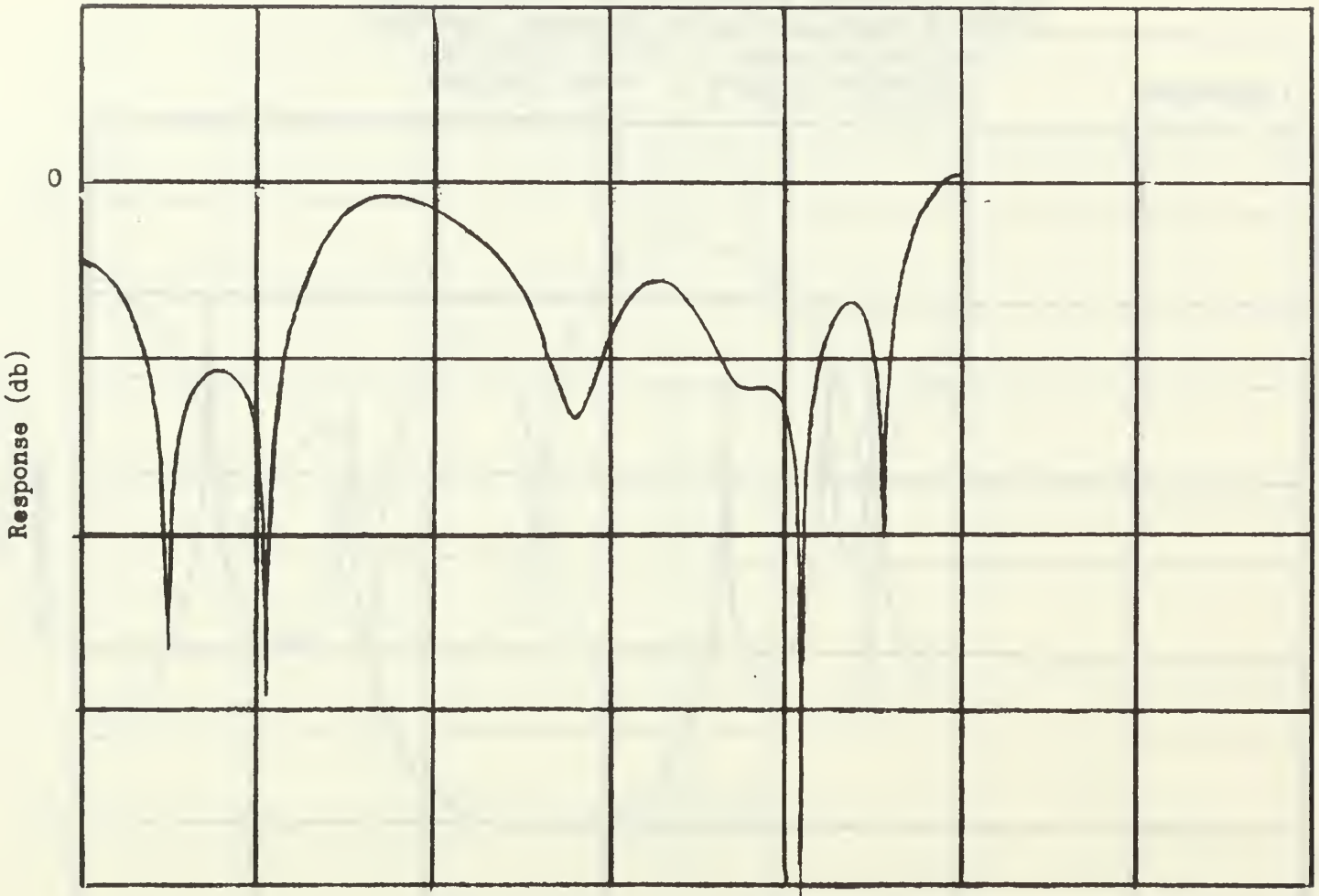
Frequency responses for $\theta = 0^{o}$ after adapting for 300 cycles of the desired frequency 500 Hz.



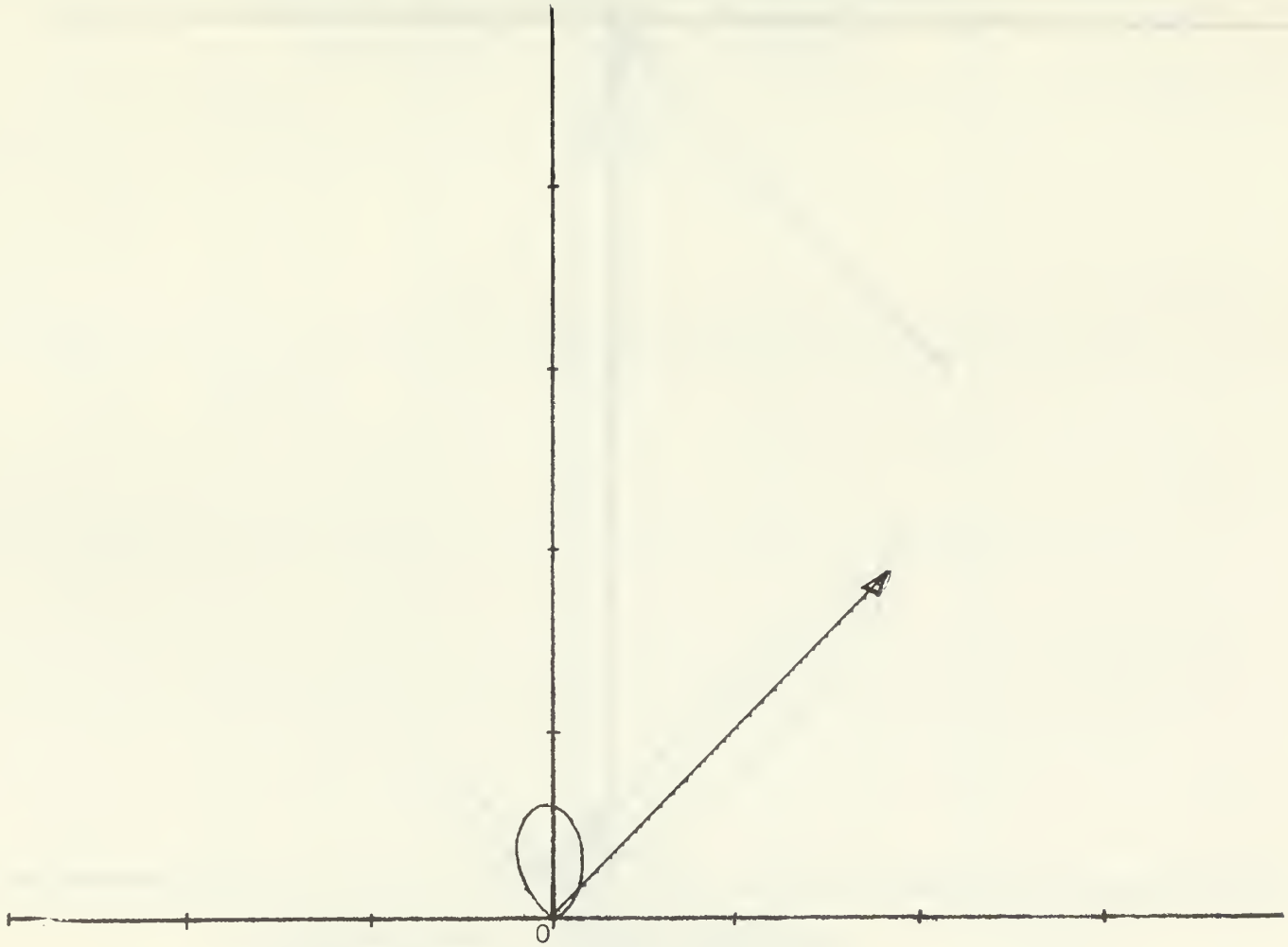
Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 30^{o}$ after adapting for 300 cycles of the desired frequency 500 Hz.



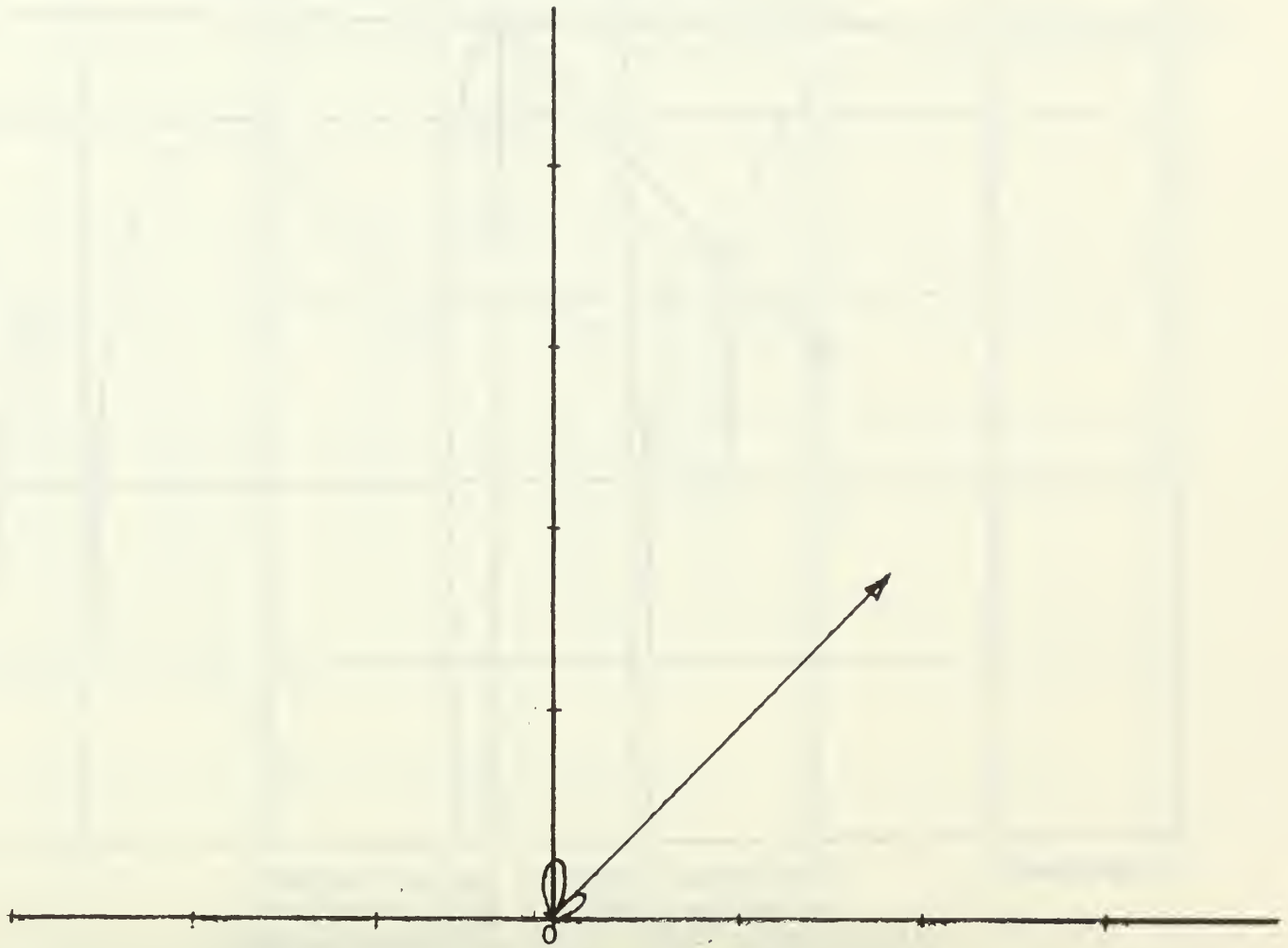
Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 45^{\circ}$ after adapting
for 300 cycles of the desired frequency 500 Hz.



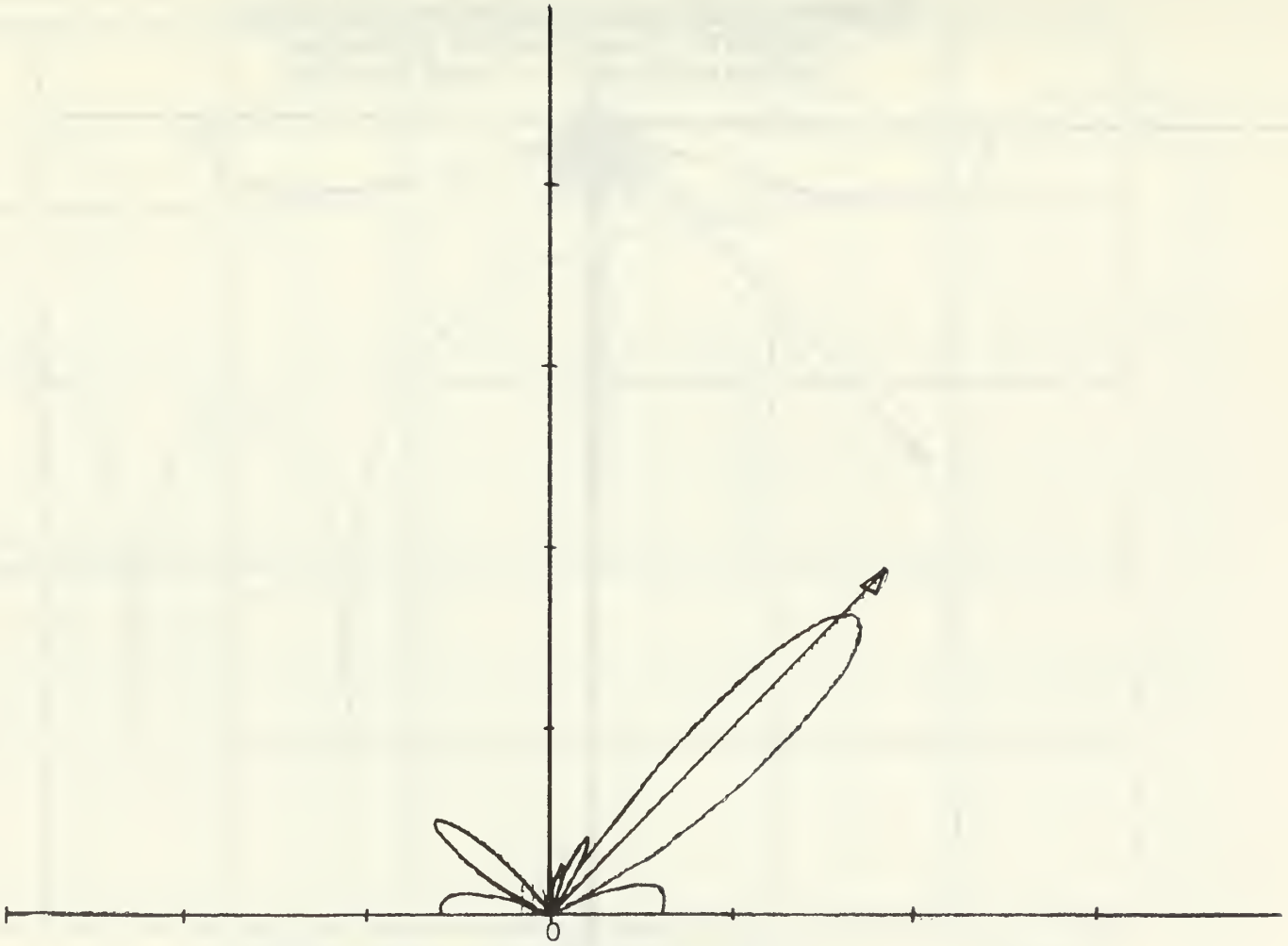
Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 88.8^{\circ}$ after adapting for 300 cycles of the desired frequency 500 Hz.



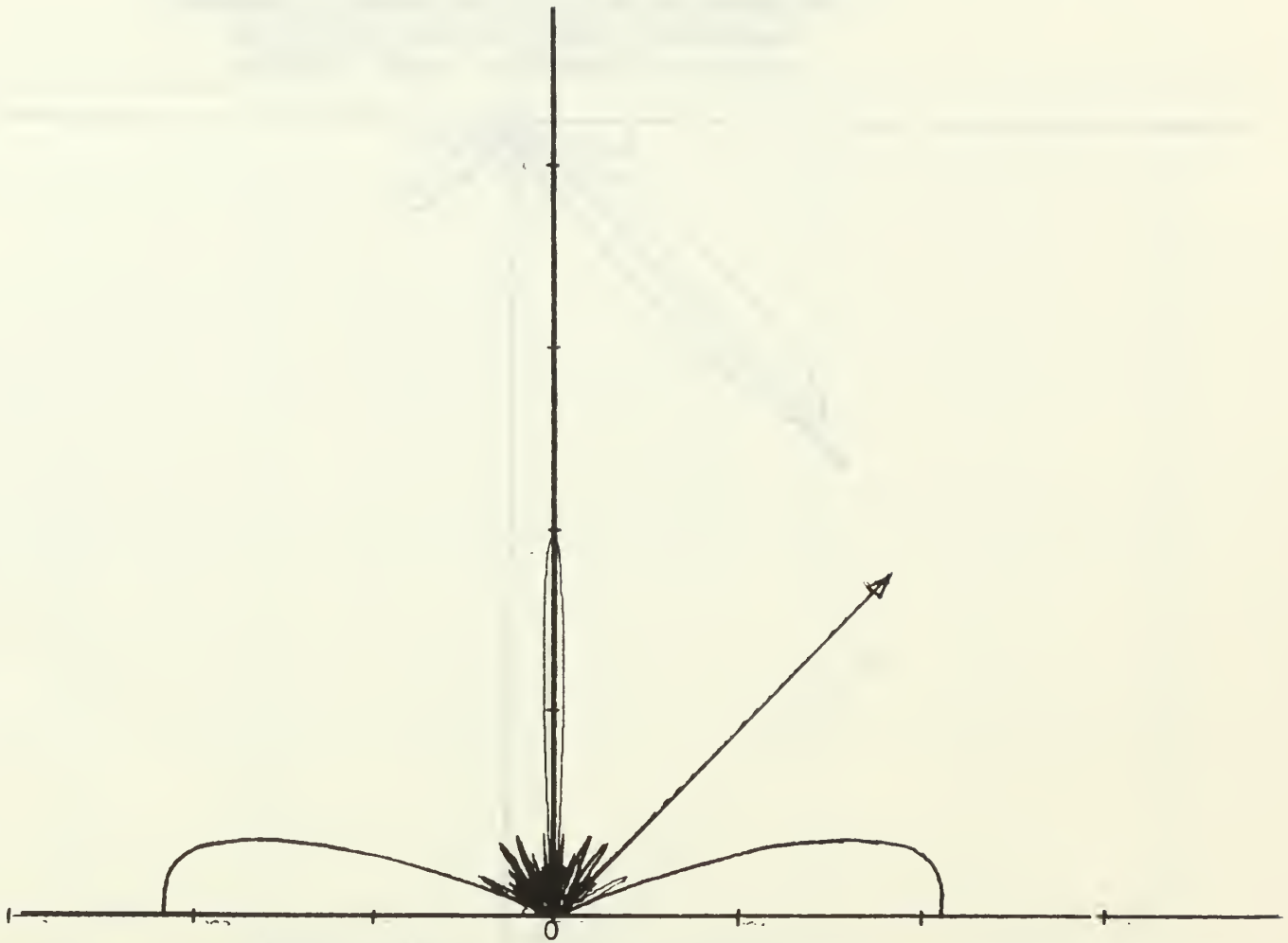
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 100 Hz after adapting
for 600 cycles of the desired frequency 500 Hz.



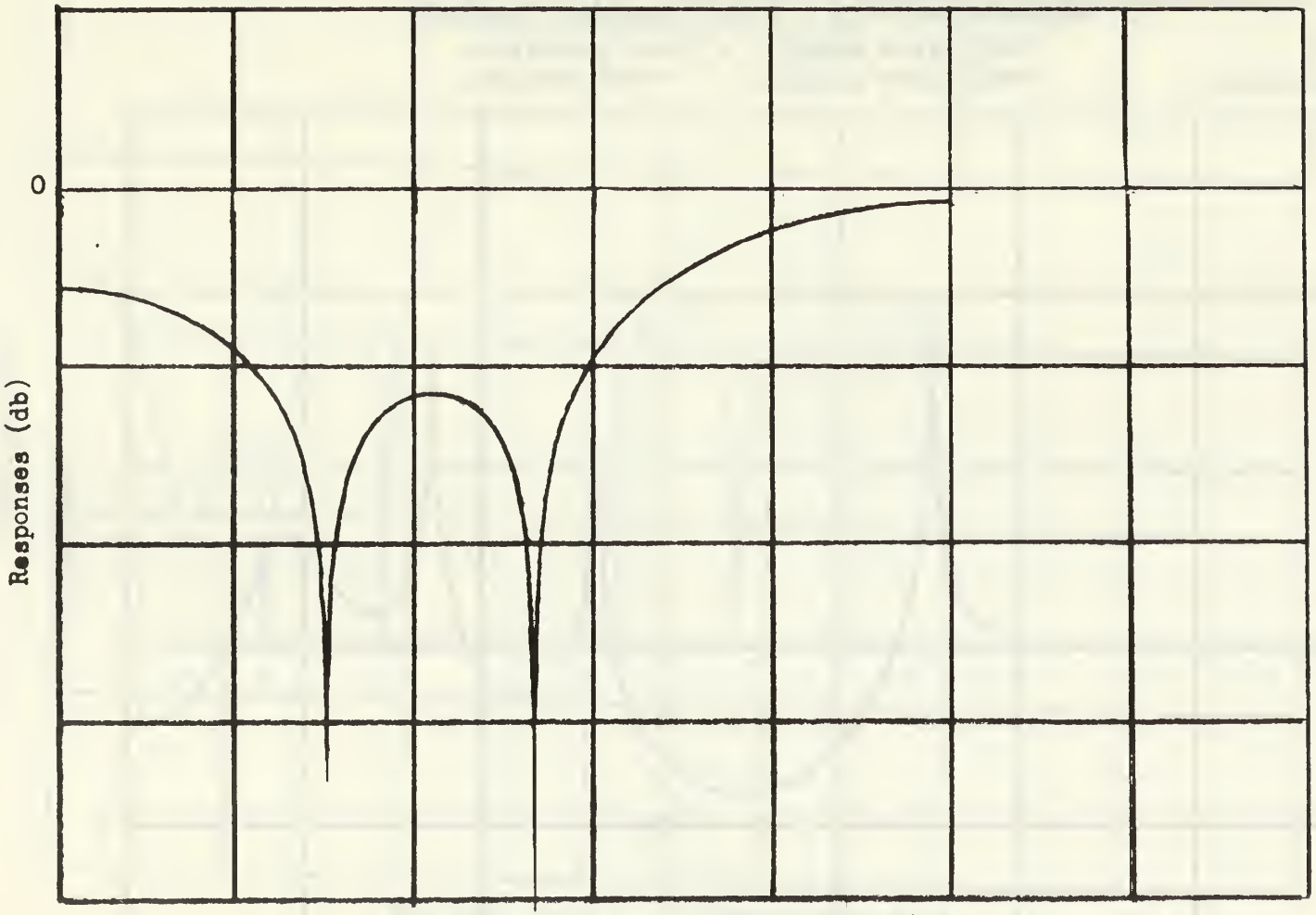
Vertical scale = 4.00E-01 units/inch
Horizontal scale = 4.00E-01 units/inch
Directivity pattern at 200 Hz after adapting
for 600 cycles of the desired frequency 500 Hz.



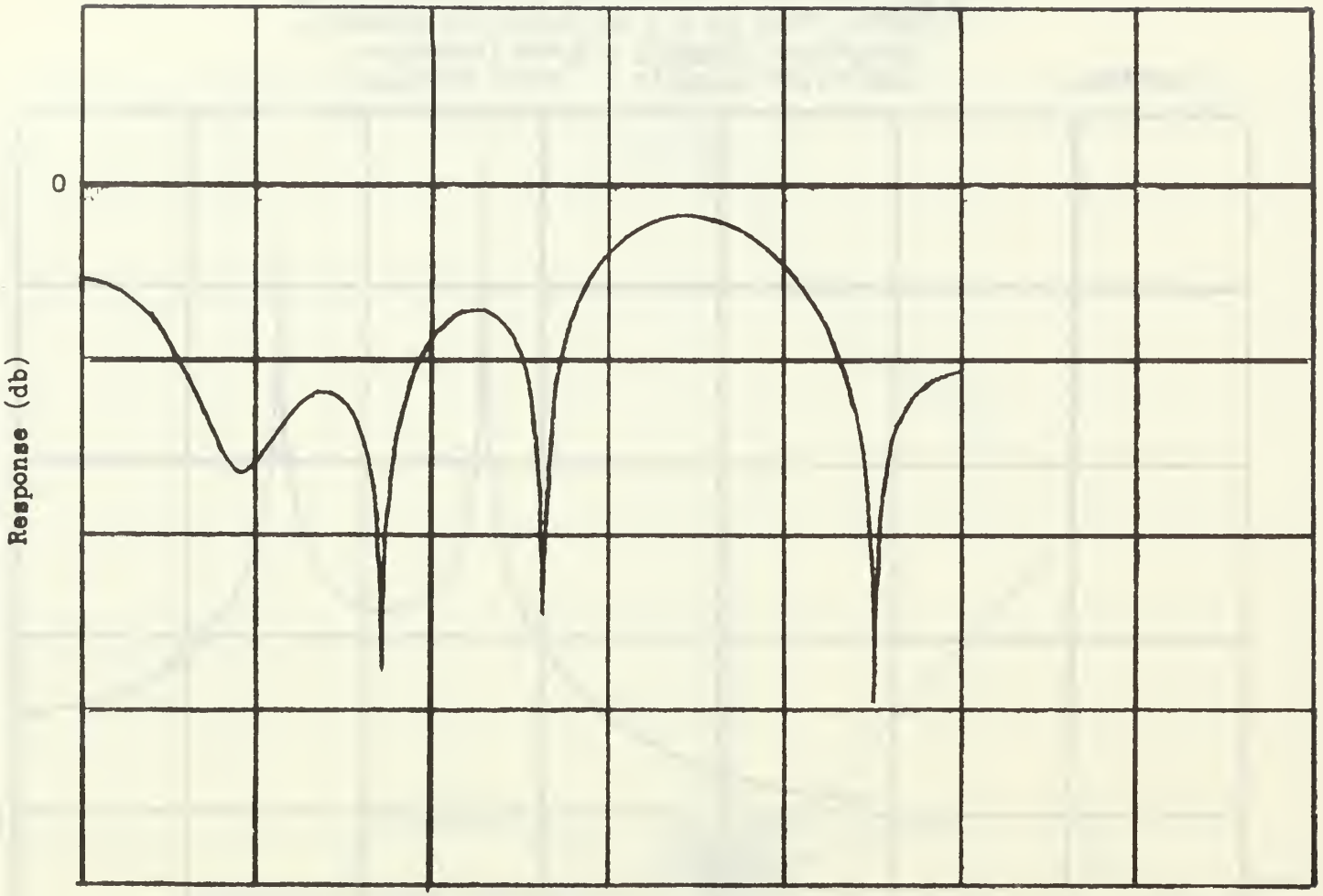
Vertical scale =4.00E-01 units/inch
Horizontal scale =4.00E-01 units/inch
Directivity pattern at 500 Hz after adapting
for 400 cycles of the desired frequency 500 Hz.



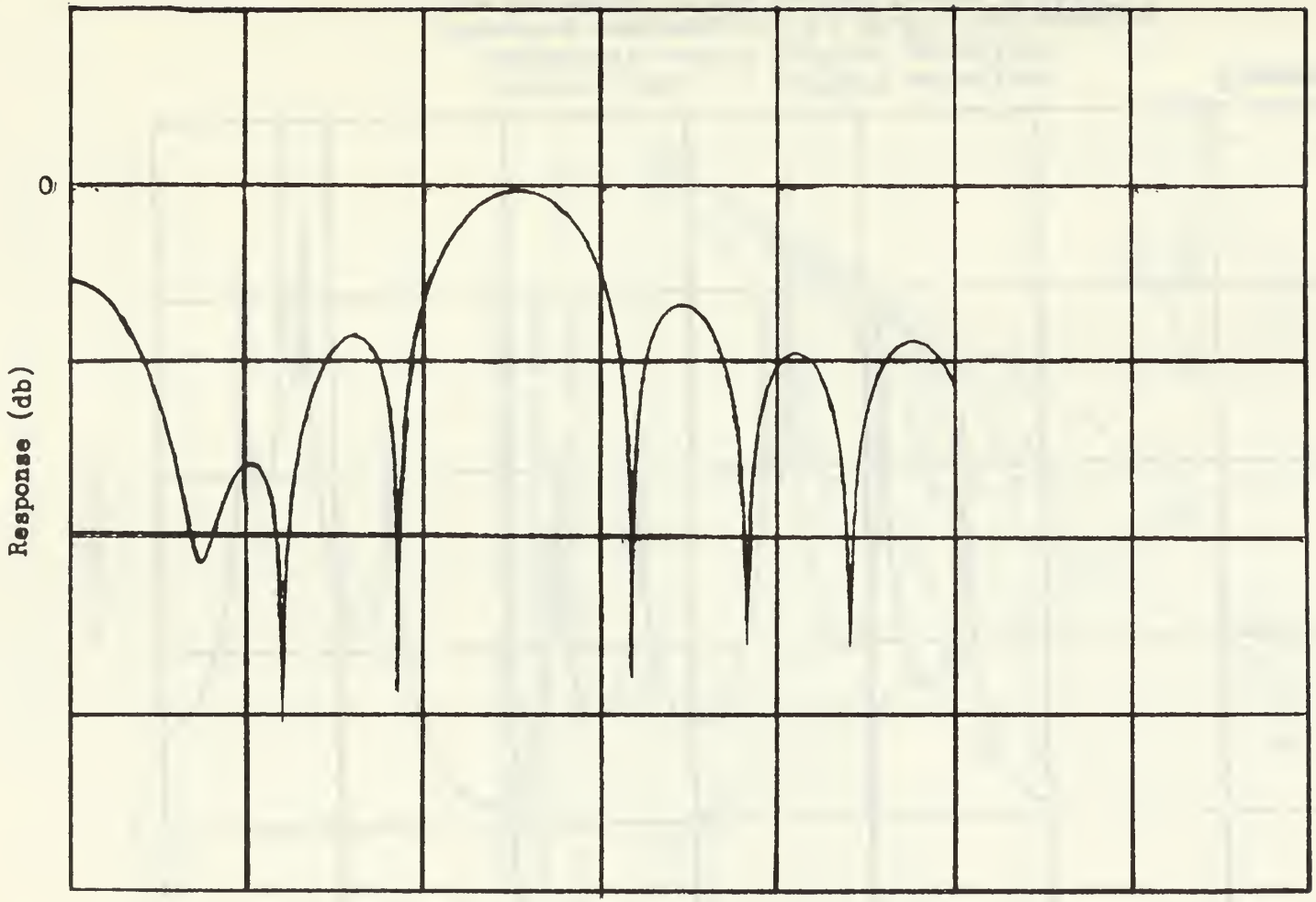
Vertical scale = 4.00E-01 units/inch
Horizontal scale=4.00E-01 units/inch
Directivity pattern at 1000Hz after adapting
for 600 cycles of the desired frequency 500 Hz.



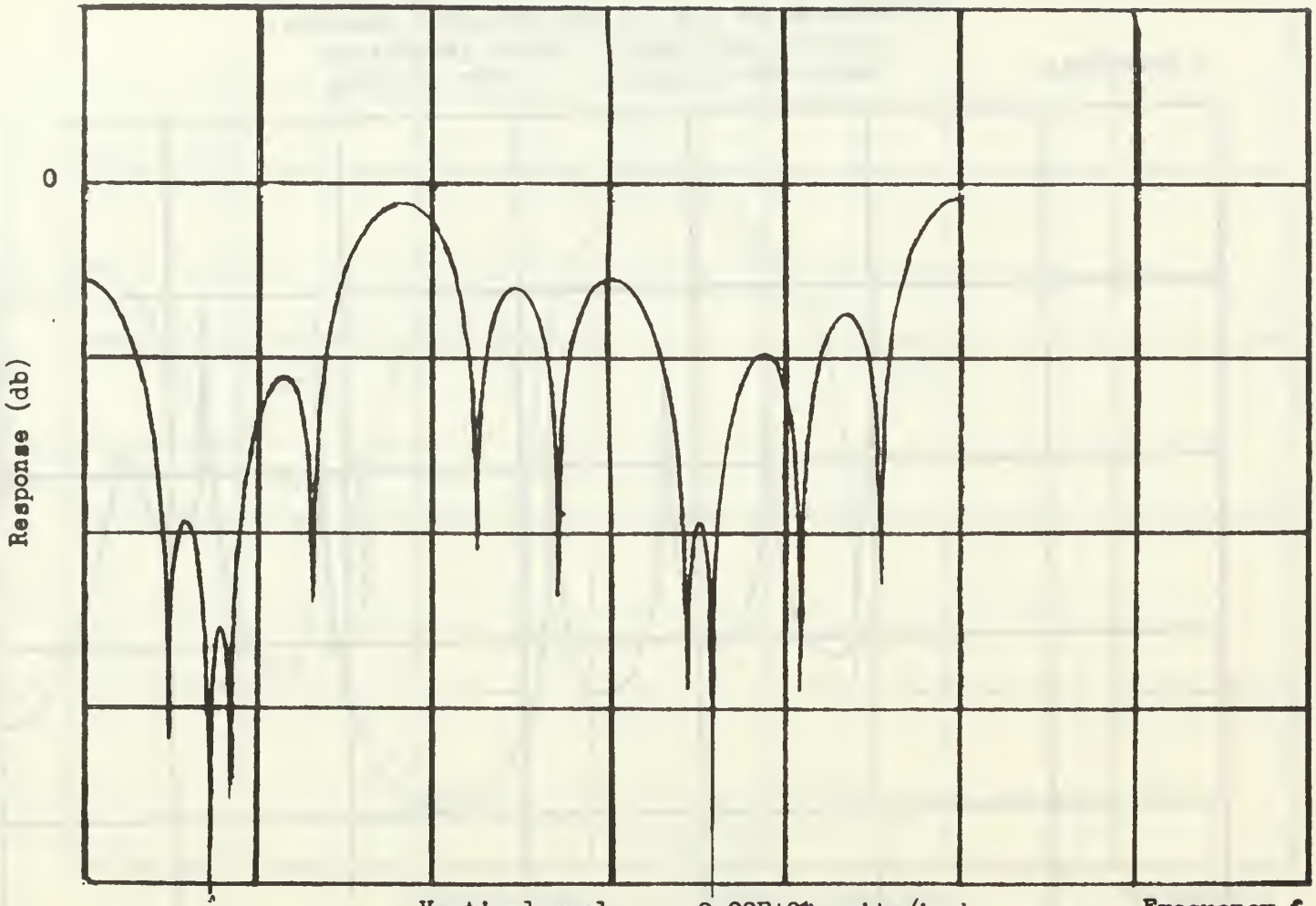
Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 0^{o}$ after adapting
for 600 cycles of the desired frequency 500 Hz.



Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 30^{o}$ after adapting for 600 cycles of the desired frequency 500 Hz.



Vertical scale = 2.00E+01 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 45^{\circ}$ after adapting for 600 cycles of the desired frequency 500 Hz.



Vertical scale = 2.00E+02 units/inch
Horizontal scale = 2.00E+02 units/inch
Frequency responses for $\theta = 88.8^{\circ}$ after adapting for 600 cycles of the desired frequency 500 Hz.

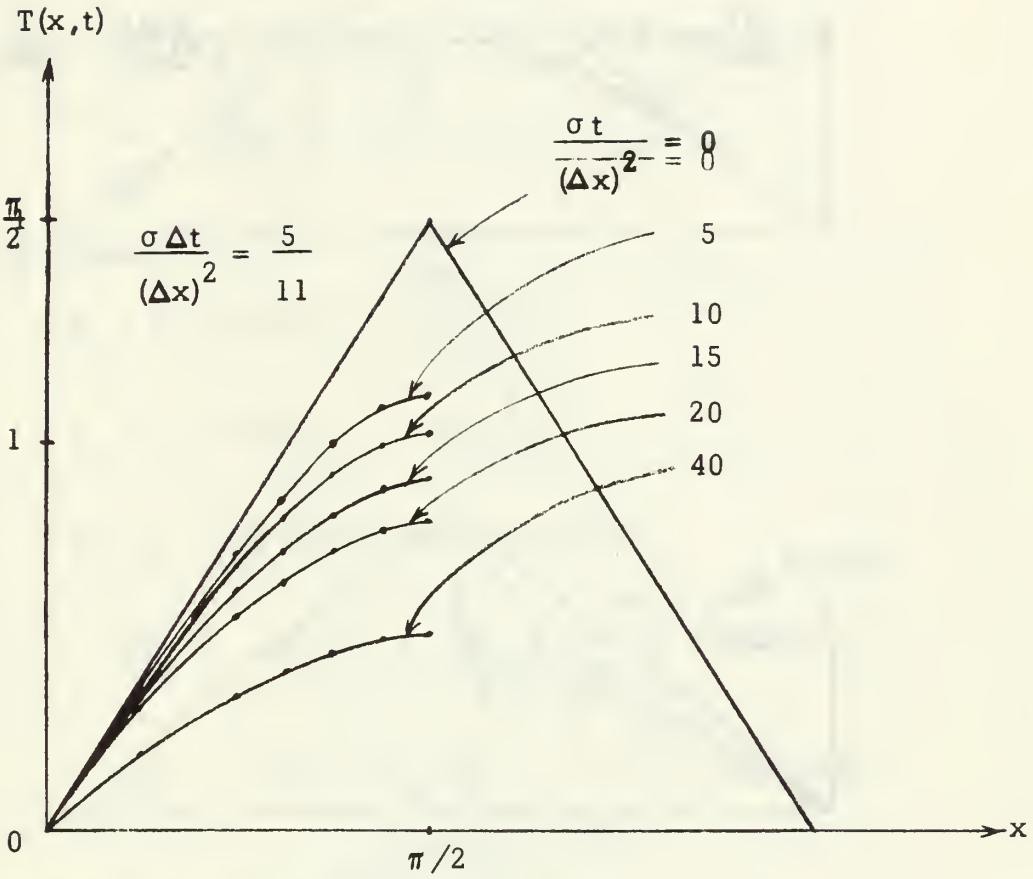
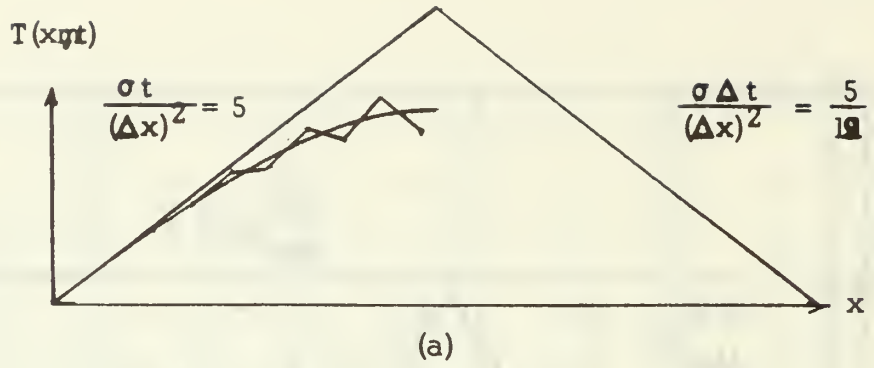
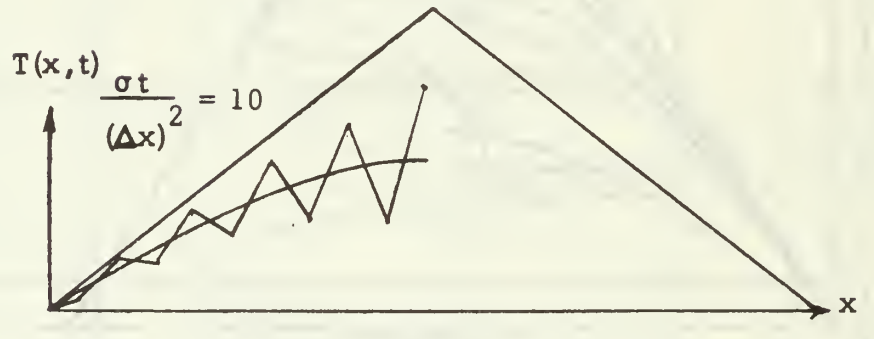


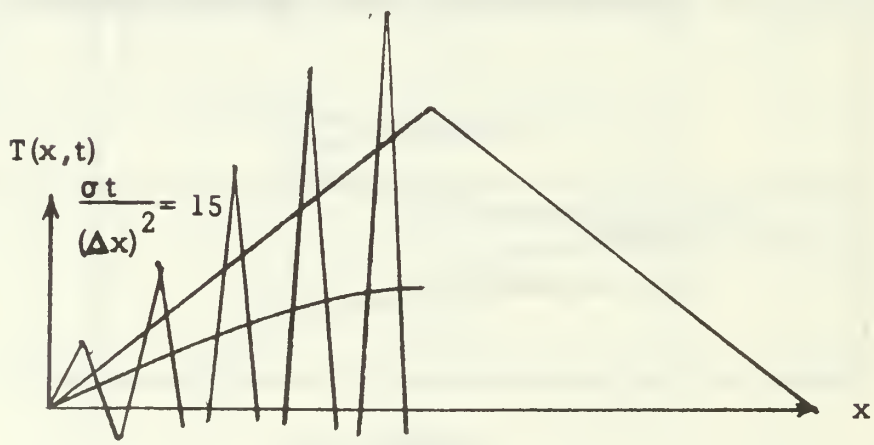
Figure 1. Diffusion equation solution by Fourier series and stable discrete difference methods. Curves = Fourier solution; dots = discrete solution.



(a)

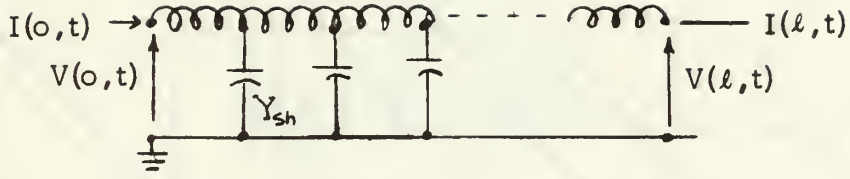


(b)



(c)

Fig. 2. Instabilities in discrete diffusion equation solution.



For general line

$$Z_s = R_s + j2\pi fL_s + \frac{1}{j2\pi fC_s}$$

$$Y_{s\ell} = G + j2\pi fC_{sh} + \frac{1}{j2\pi fL_{sh}}$$

V(x,t), I(x,t) satisfy instantaneously

$$\frac{\partial V}{\partial x} = -\left(R_s I + L_s \frac{\partial I}{\partial t} + \frac{1}{C_s}\int^t I(x,\xi)d\xi\right)$$

$$\frac{\partial I}{\partial x} = -\left(GV + C_{sh}\frac{\partial V}{\partial t} + \frac{1}{L_{sh}}\int^t V(x,\xi)d\xi\right)$$

or for steady state $e^{j2\pi ft}$ variations

$$\frac{d\hat{V}(x)}{dx} = -Z\hat{I}(x)$$

$$\frac{d\hat{I}(x)}{dx} = -Y_{sh}\hat{V}(x)$$

where

$$V(x,t) = \mathrm{Re}\left(\hat{V}(x)e^{j2\pi ft}\right)$$

$$I(x,t) = \mathrm{Re}\left(\hat{I}(x)e^{j2\pi ft}\right)$$

Fig. 3. The One-Dimensional Transmission Line Equations

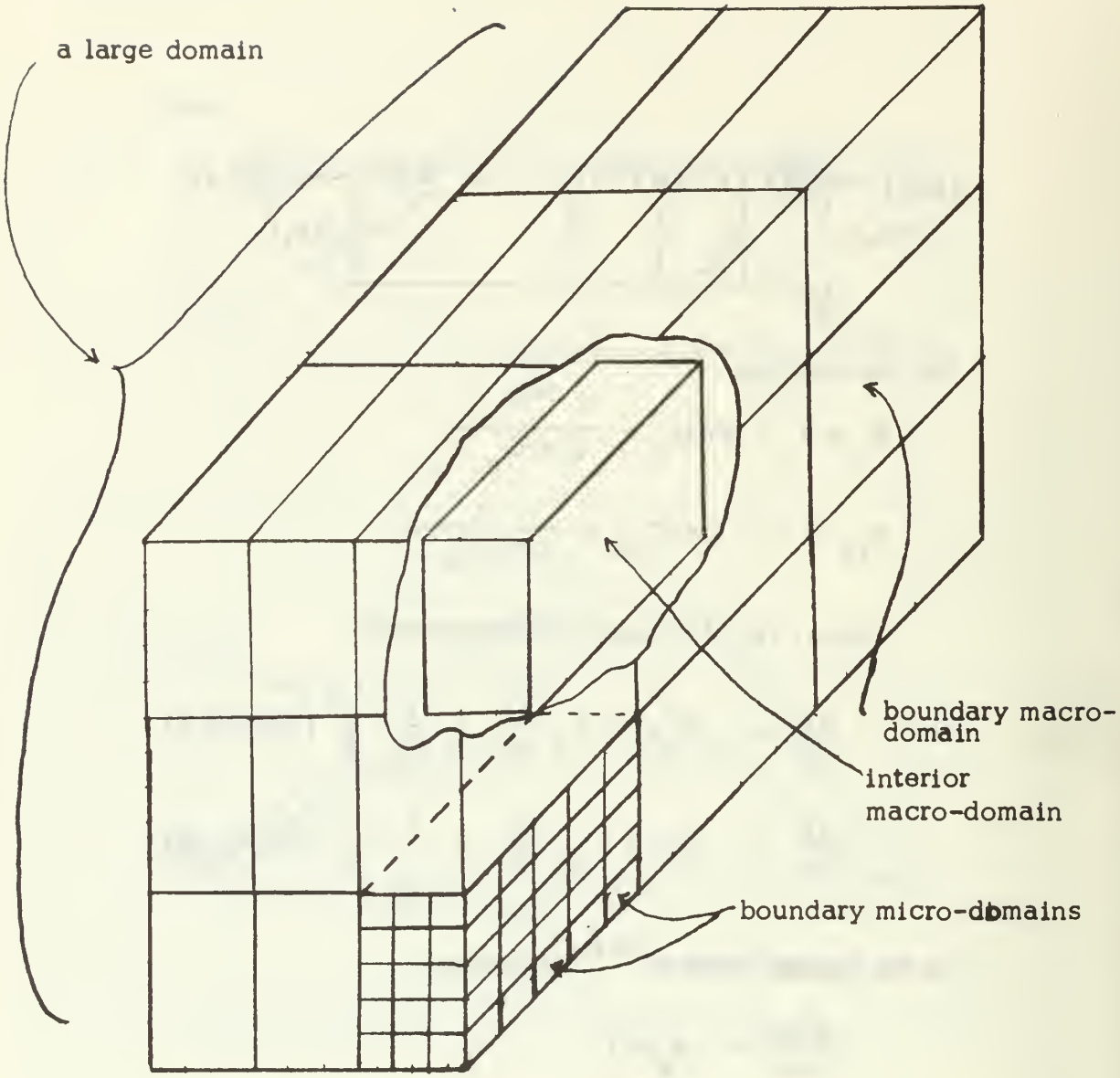


Fig. 4. Geometric model of the integration domains

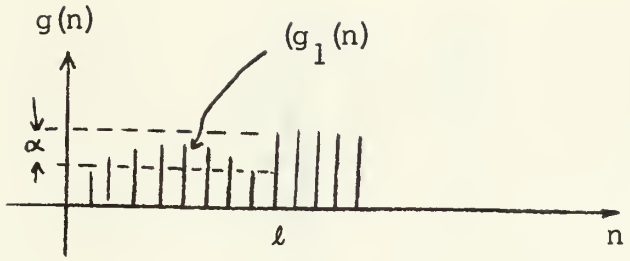
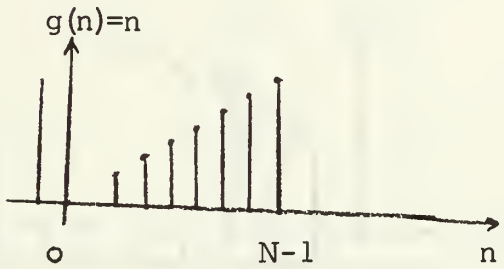
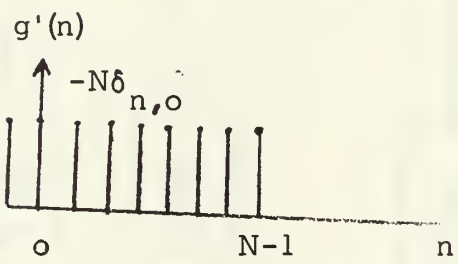


Fig. 5. Function having discontinuity at $n = \ell$.



(a)

function g(n) = n



(b)

Fig. 6. Derivative of function g(n) = n

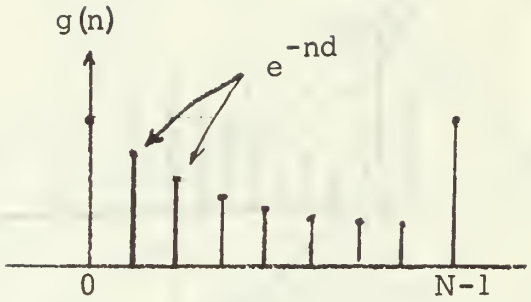


Fig. 7. Recursive example: The exponential function

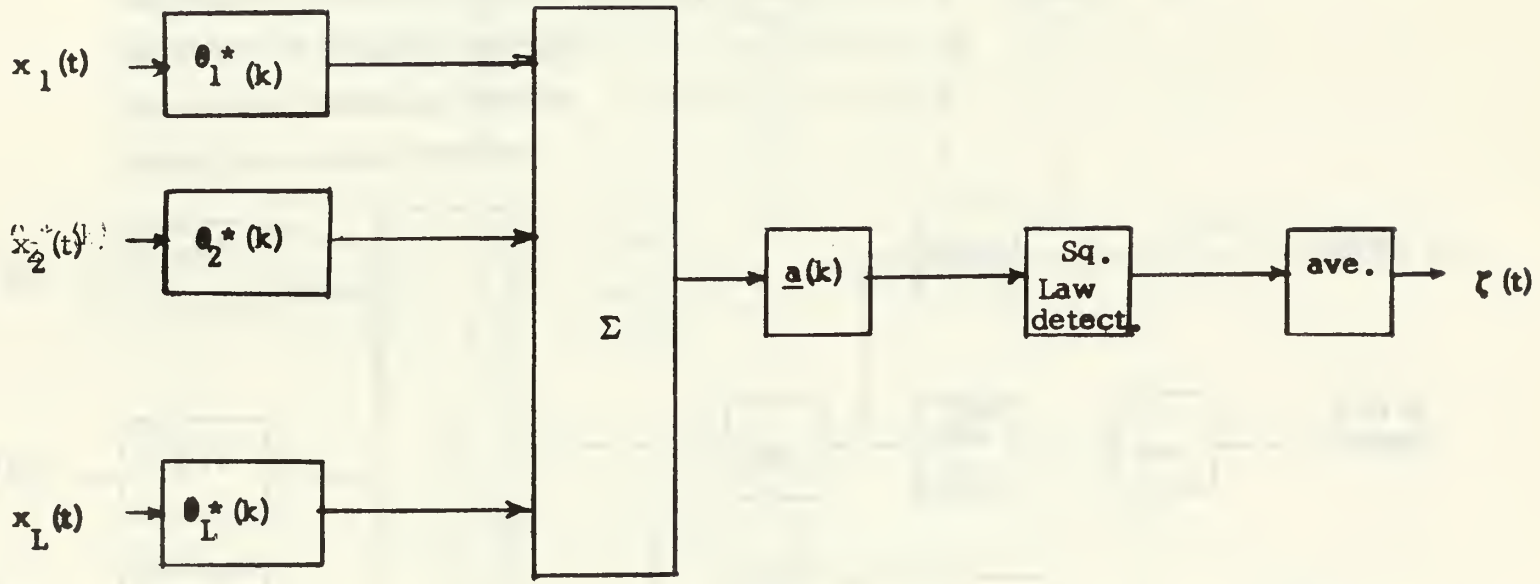
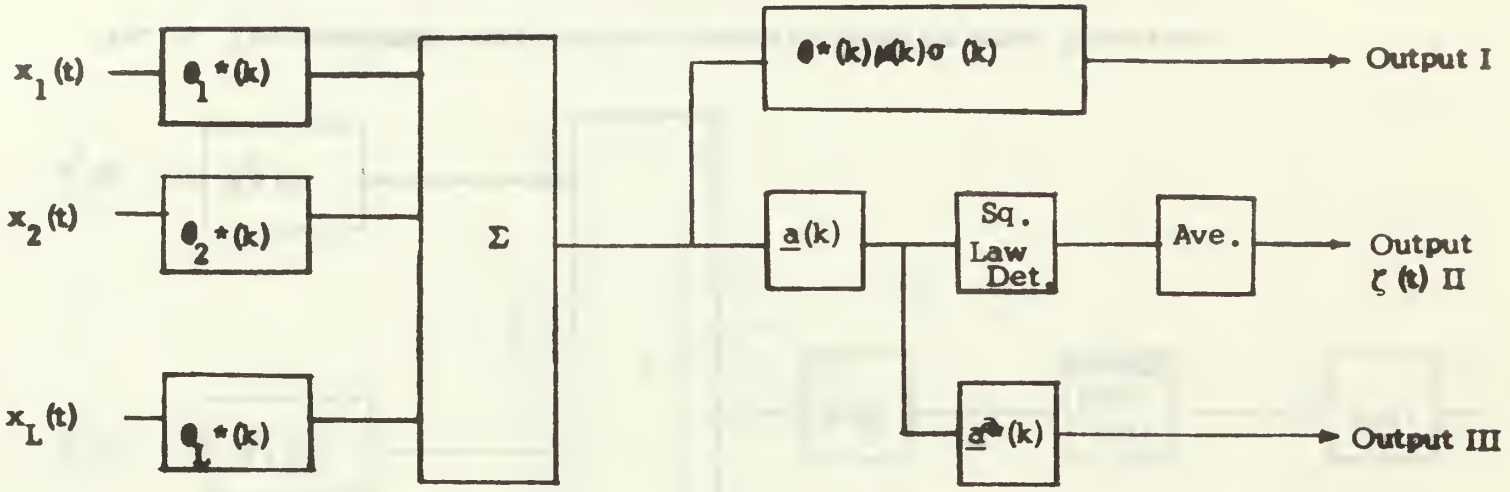


Fig. 9. The quadratic-cost-function optimal-estimator Array Processor.



DETECTION KNOW SIGNAL ---------- I

DETECTION RANDOM SIGNAL ---------- II

ESTIMATION RANDOM SIGNAL ---------- III

DETECTION RANDOM SIGNAL (MAX d) ---------- II $|\underline{a}(k)|^2 \rightarrow p_k$

DETECTION UNKNOWN SIGNAL ---------- II $\rightarrow o$

ESTIMATION UNKNOWN SIGNAL ---------- III $p_k^{-1} \rightarrow o$

Fig. 10. The general quadratic-cost-function optimal-estimator array processor.

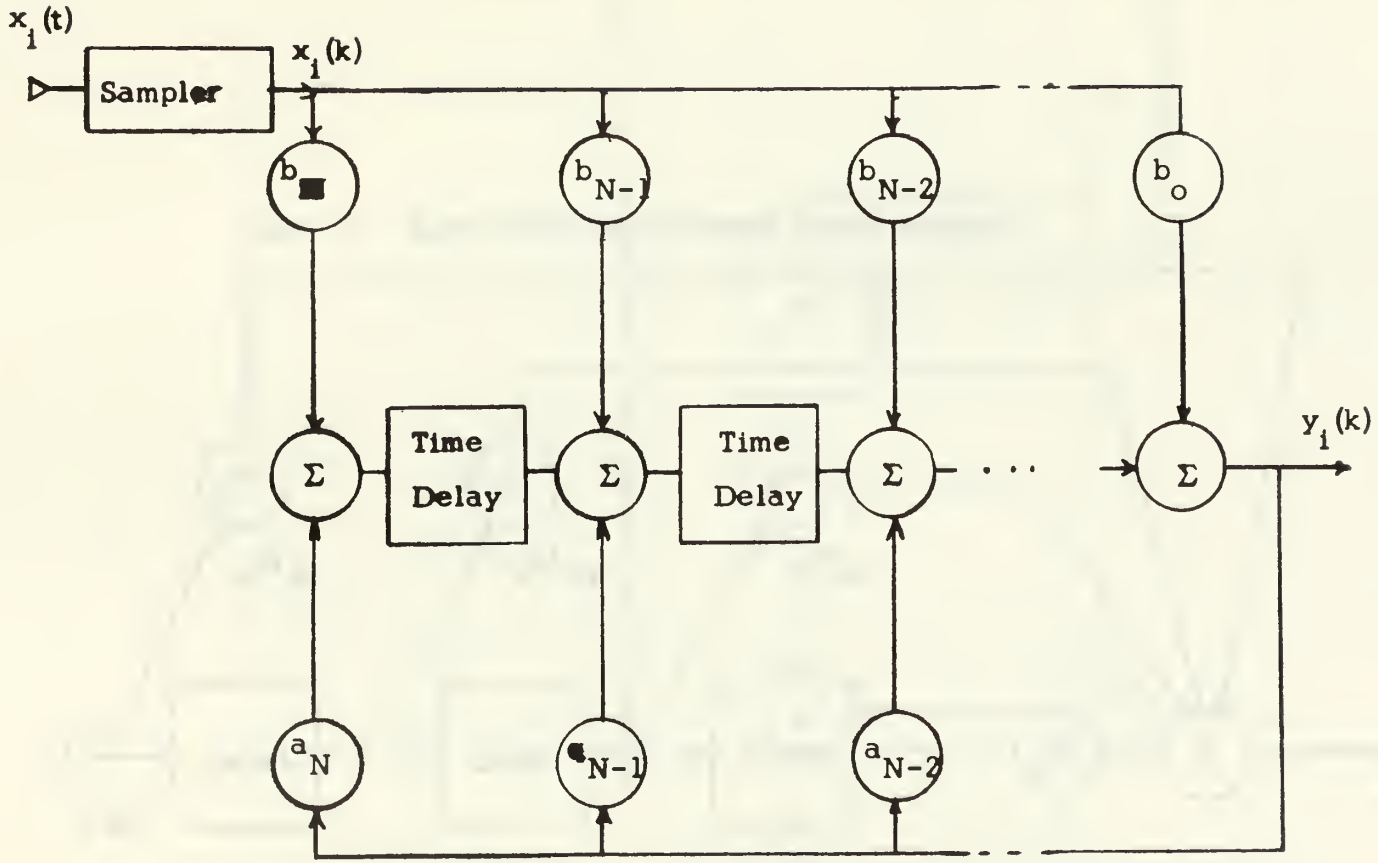


Fig. 11. General feedback, feed-forward sampled-data network

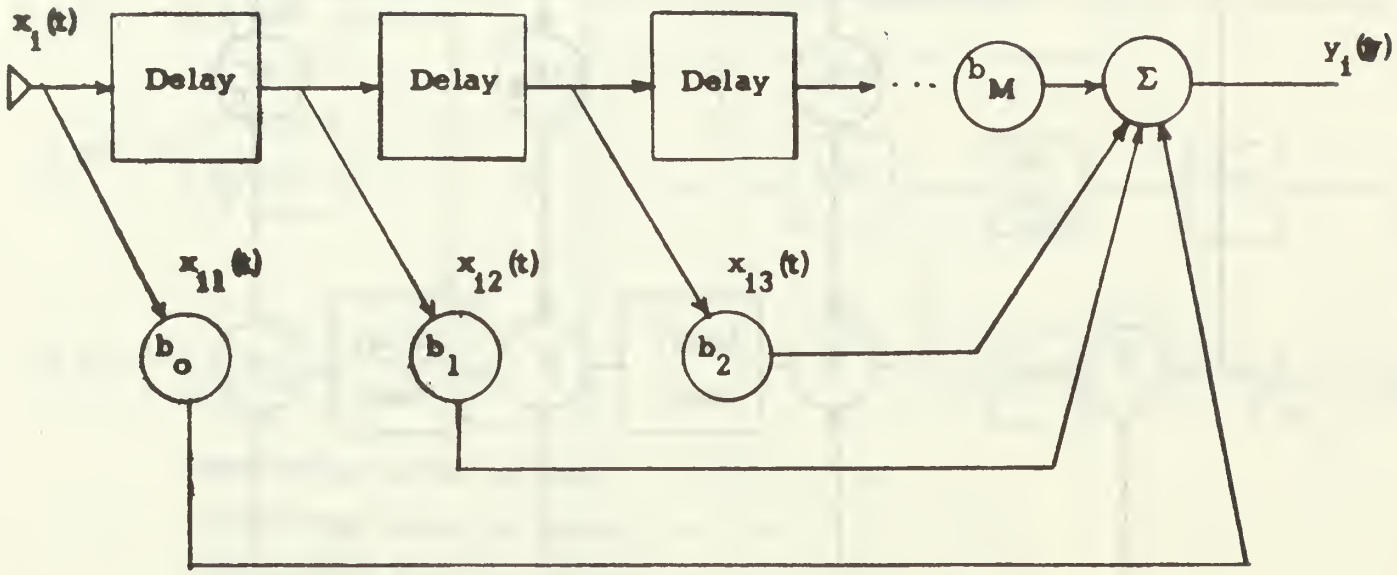


Fig. 12. Wide-band feed-forward delay network.

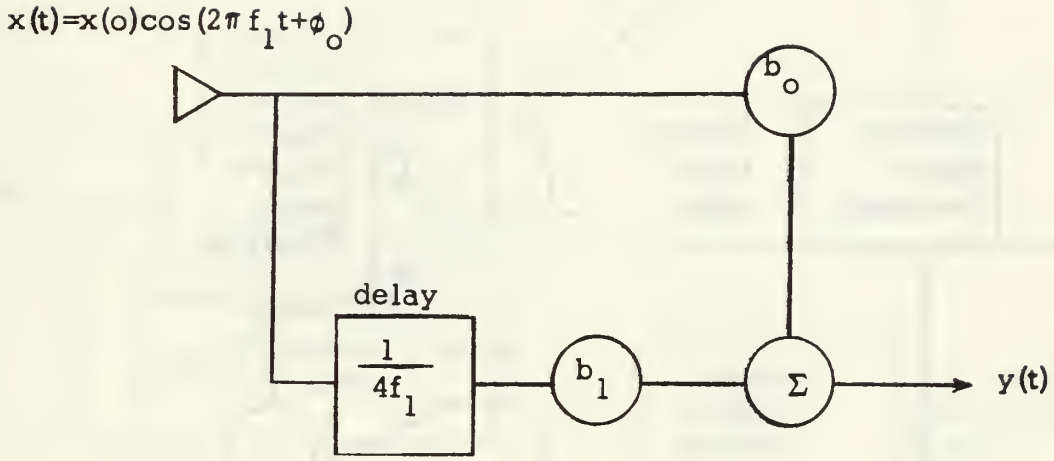


Fig. 13. Narrow-band feed-forward delay network

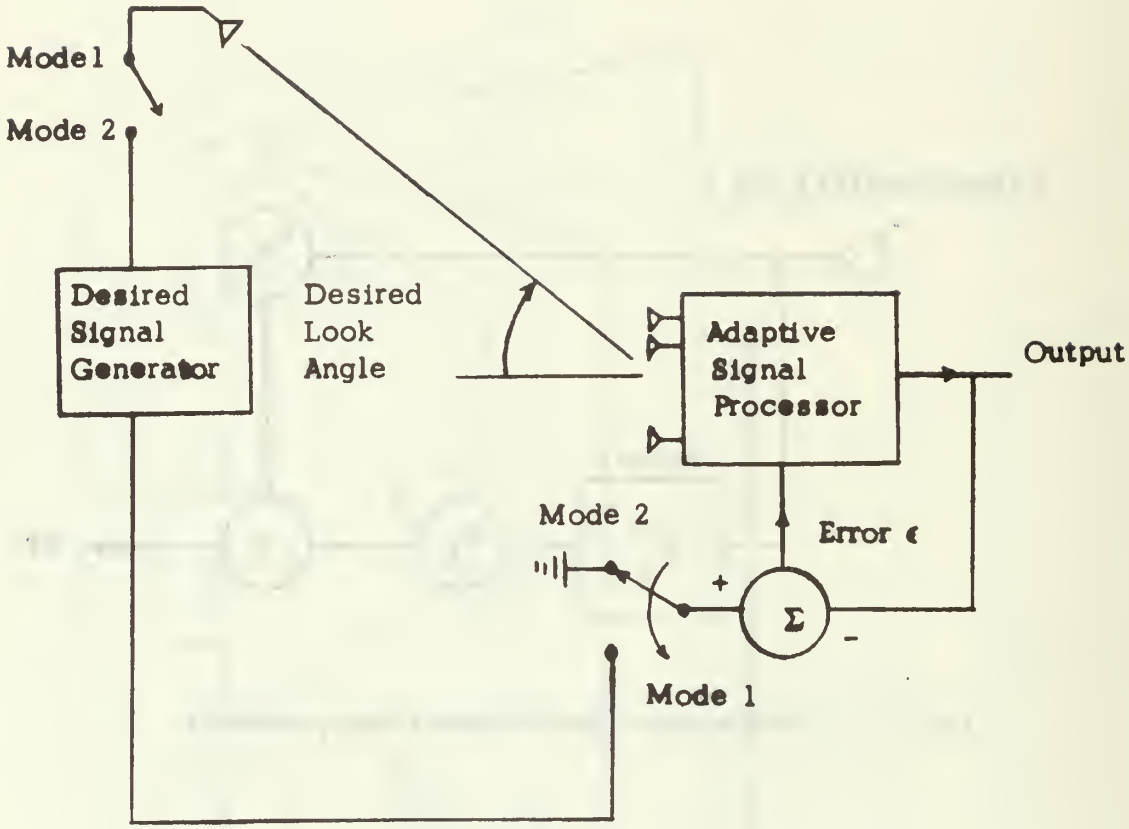


Fig. 14. The on-off adaptive network scheme

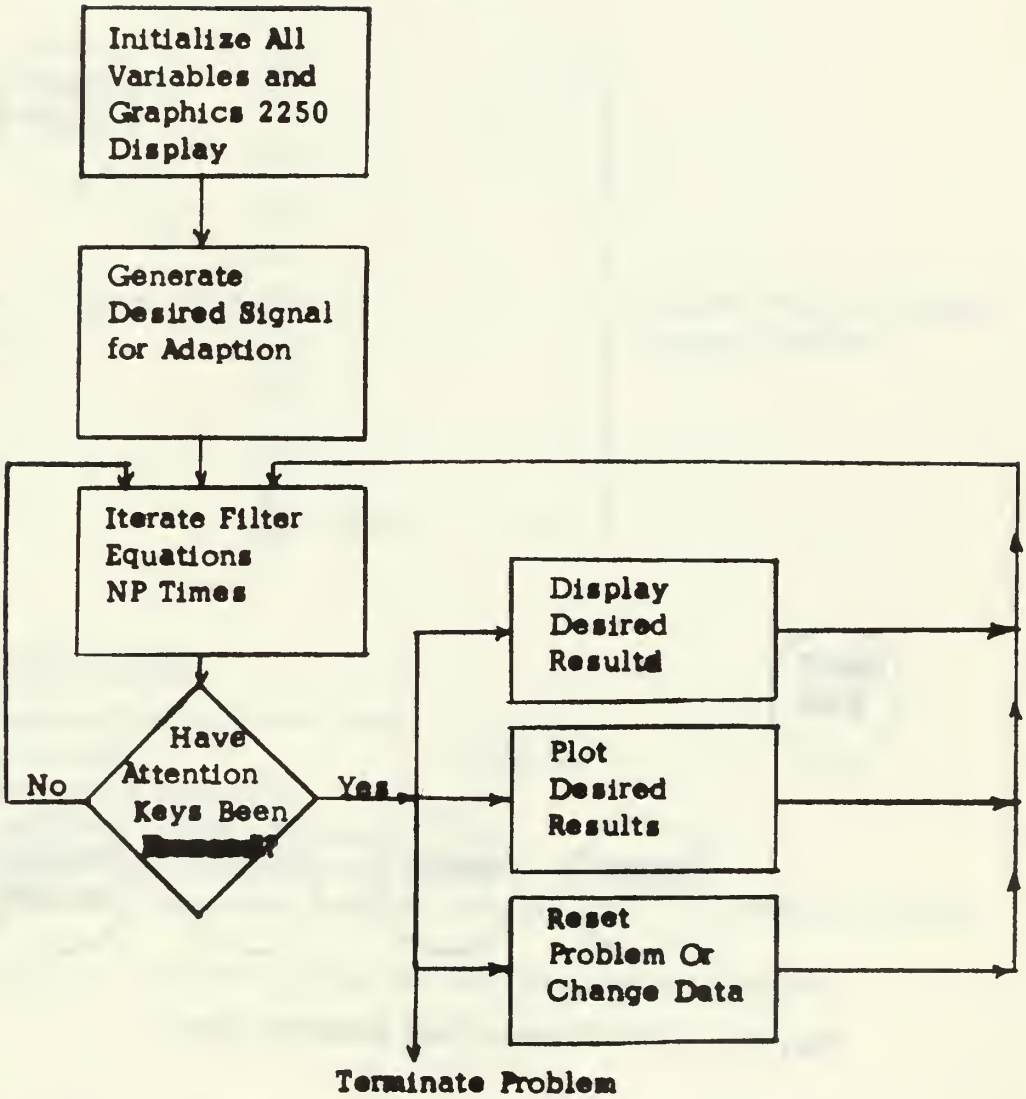


Fig. 15. Flow Diagram of Array Simulation Program.

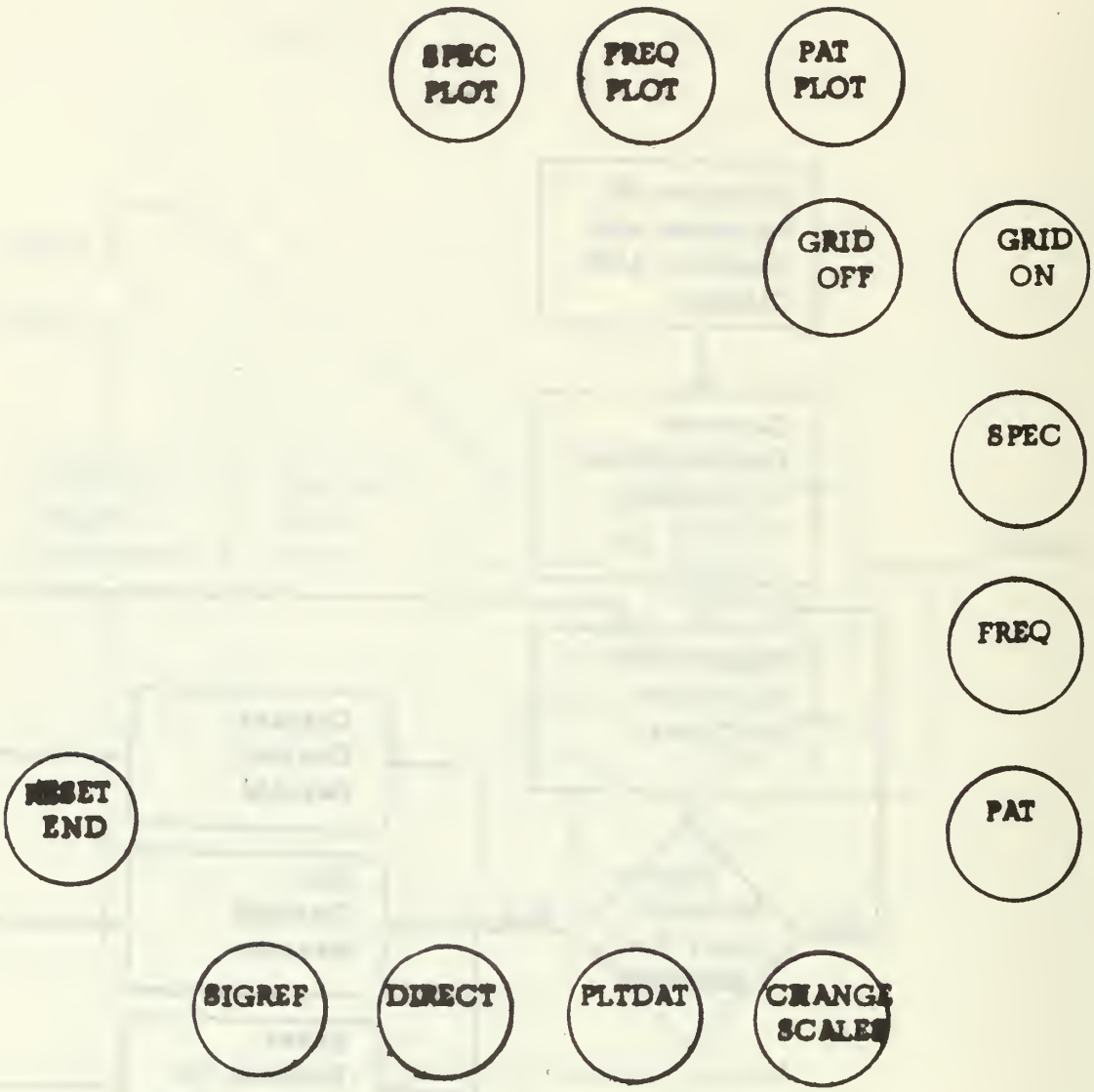
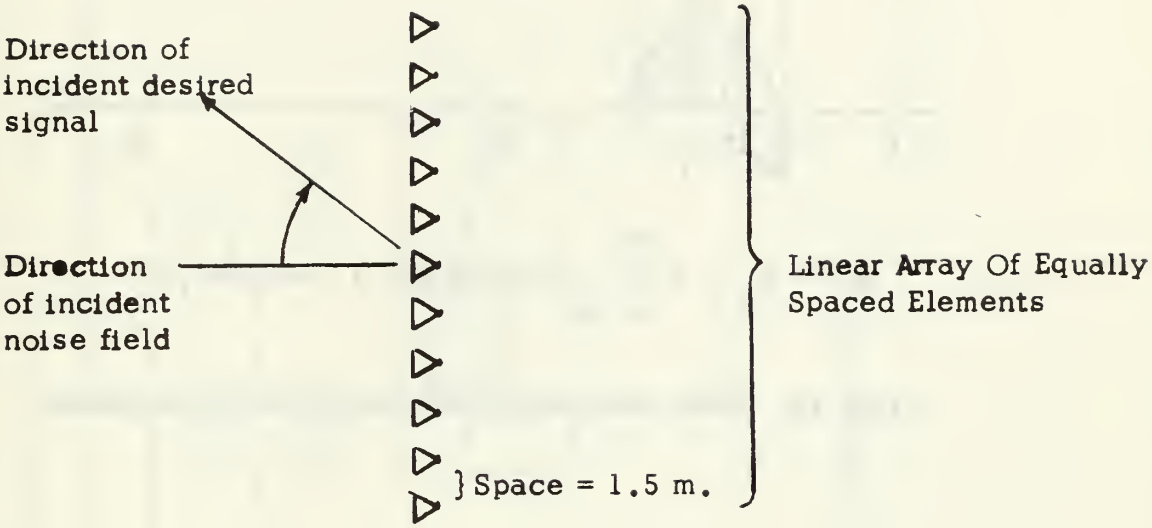


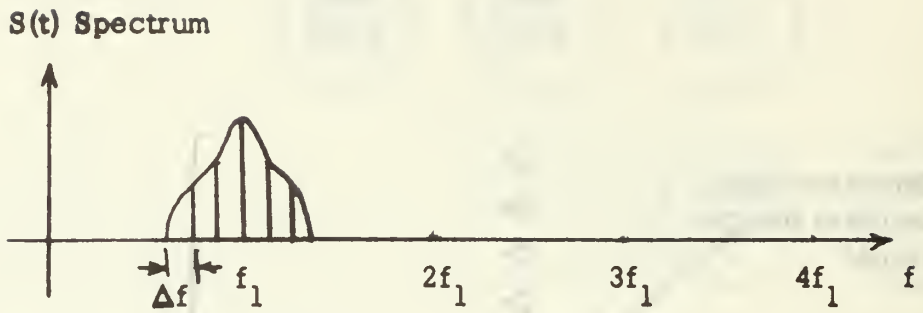
Fig. 16. 2250 Display Unit Attention Keys.



Problem Data:

Desired Signal Center Freq. = 500 Hz
Bandwidth = 200 Hz
Number of Elements = 11
Number of Delays/Element = 2
Pattern Plots at Angles of o°, 30°, 45°, 90°
Frequency Response Plots at 100 Hz, 250 Hz, 500 Hz, 1000 Hz

Fig. 17. Physical Layout of Array Simulation Program.



$$S(t) = A_o \left[1 + \sum_{k=-M}^{+M} \cos(2\pi (k\,\Delta f)\, t)\right] \cos(2\pi f_1 t)$$

Fig. 18. Wide-band Amplitude Modulated Input Signal.

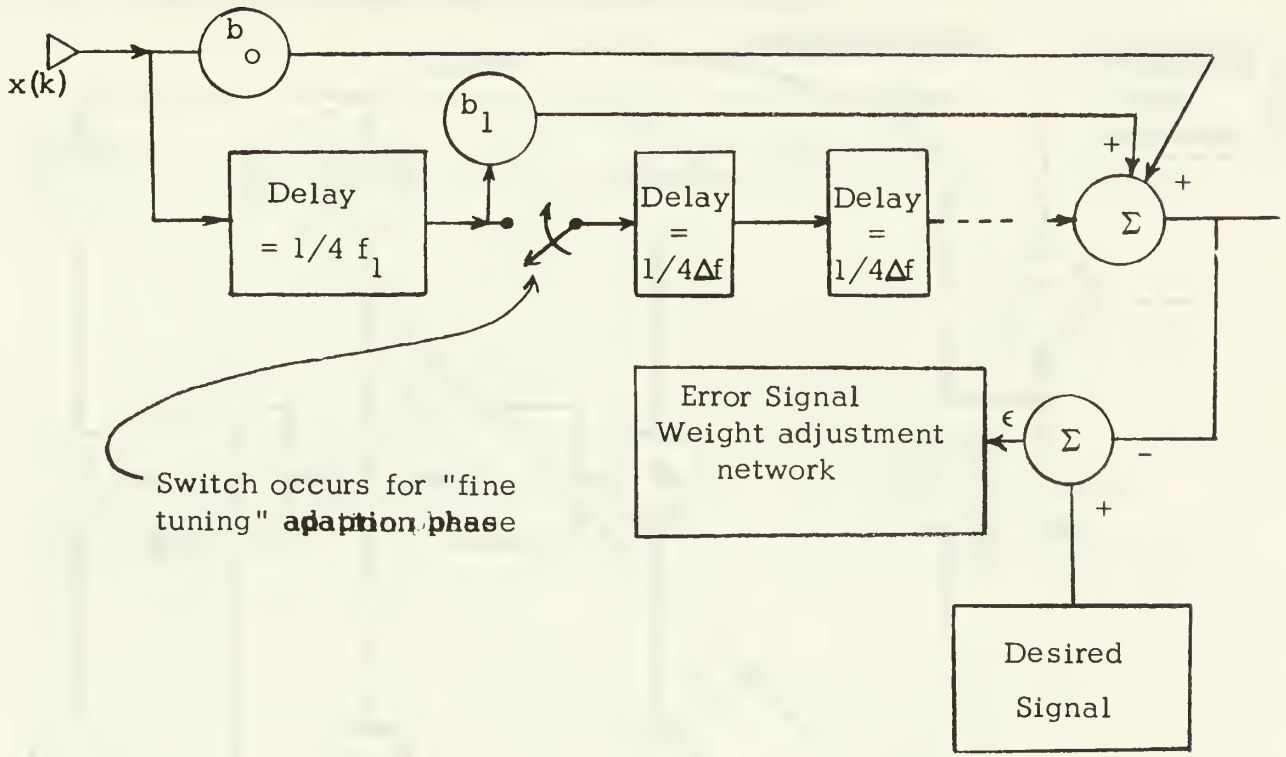


Fig. 19. Adaption network with variable delays.

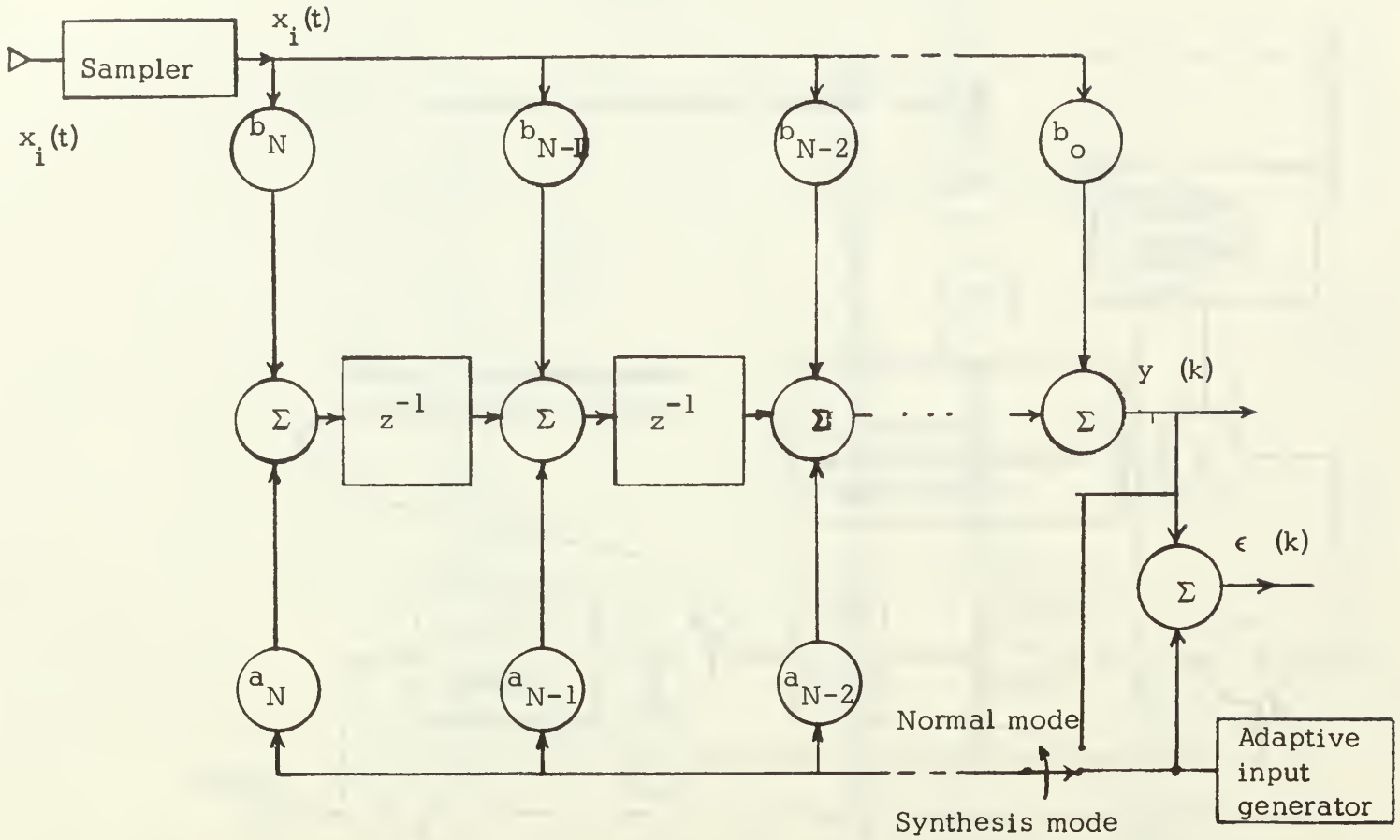


Fig. 20. Mantey's modified adaptive feedback network.

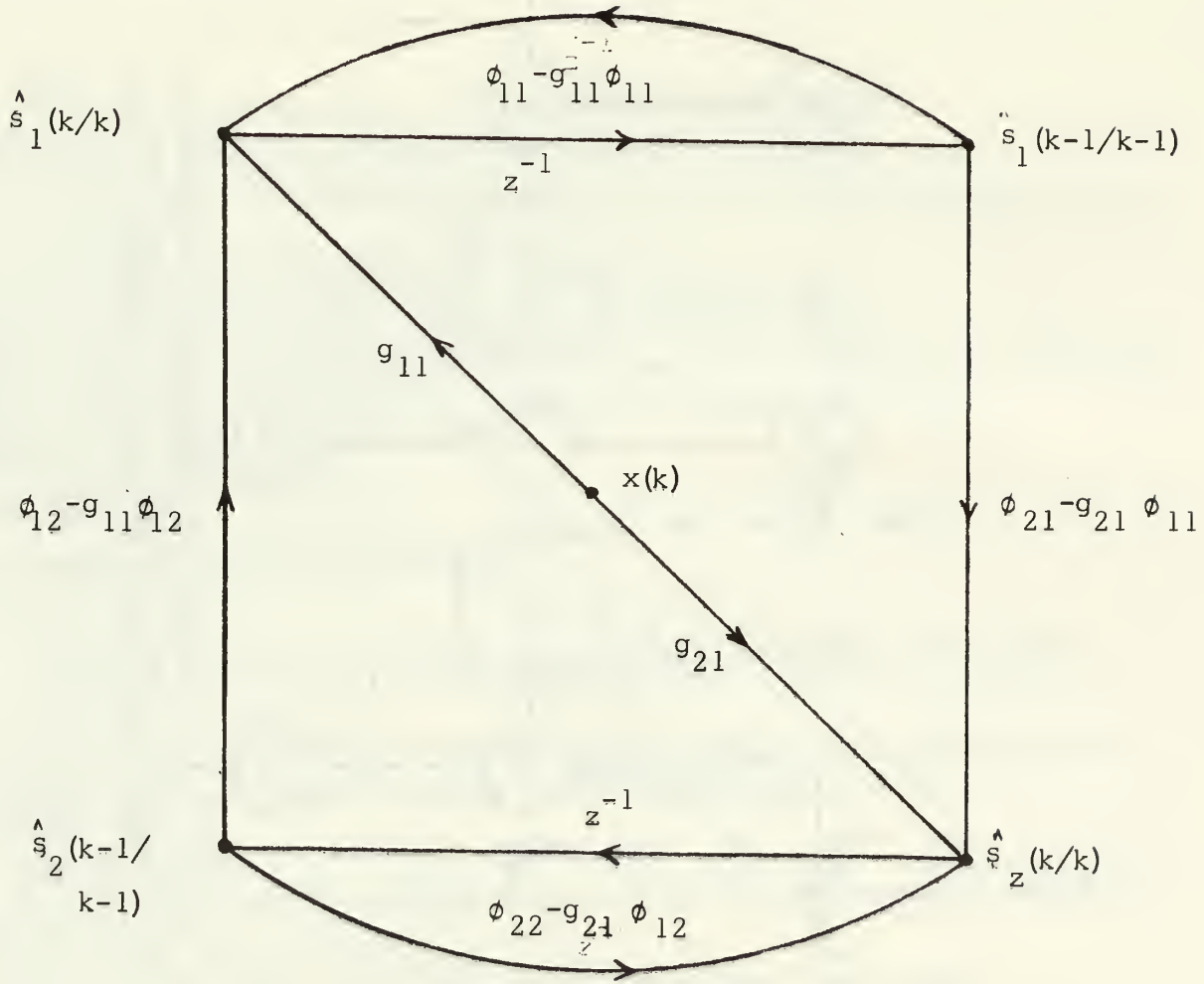


Fig. 20. Two-delay Kalman Filter State Transition Diagram.

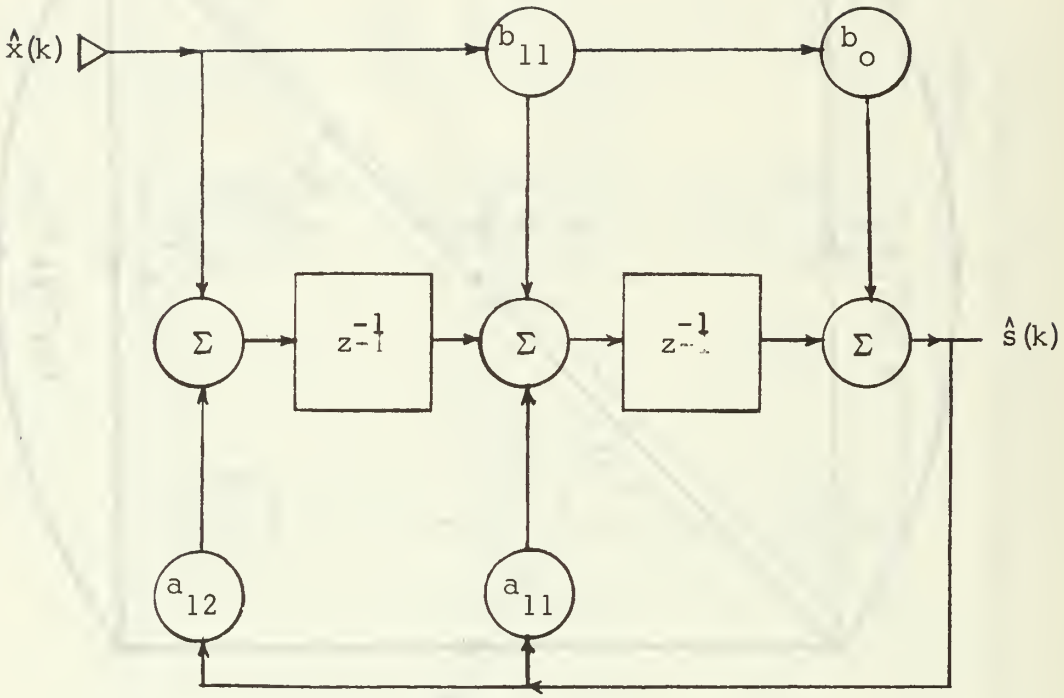


Fig. 22. Two-delay Kalman Filter Adaptive network.

```
C WV2525 INTEGRATES THE WAVE EQUATION IN TWO DIMENSIONS WITH GRAPH OUTPUT


      BLOCK DATA
      REAL   T/0.0/,DT,DT1/0.0005 /,DT2/0.0005/,DT3/0.0005/
      COMMON/TDT/T,DT,DT1,DT2,DT3
      INTEGER NX/20/,NY/25/,NT/ 500/,IIS1/100/,IIS2/200/
      COMMON/SIZE/NX,NY,NT,IIS1,IIS2
      REAL R/0.0/,LUO/6.66667E-04/,LU(25),G/0.0/,CO/1.333333E-04/,C(25),
     &DX/5.0   /,DY/5.0    /
      COMMON/PARAM/R,LUO,LU,G,CO,C,DX,DY
      REAL EX/8.0/,EY/8.0/,SLOPE/0.707/,FACT/1.0/
      COMMON/SCALE/EX,EY,SLOPE,FACT
      INTEGER IU/0/,IR/0/,IXL/2/,IYL/2/,IW/6/,IH/4/,IG/0/
      COMMON/DRW/IU,IR,IXL,IYL,IW,IH,IG
      INTEGER IWRT/100/,IDRW/100/,IEN/  5/,IANL/100 /,ISAMP/16/,IDFT/100
     &/,NWV/1 /,JDET/20/,ITYPE/1/
      COMMON/PGMC/IWRT,IDRW,IEN,IANL,ISAMP,IDFT,NWV,JDET,ITYPE
      INTEGER II/1/,NARB/0 /,NARE/25/
      COMMON/STF/II,NARB,NARE
      REAL DV(25,25,3)/1875*0.0/
      COMMON/DERIV/DV
      END                                    <- END OF BLOCK DATA
      COMPLEX*8DA(10,16)/160*(0.0,0.0)/
      REAL*8ITI(10,12)
      REAL LUO
      REAL LU(25),C(25)
      REAL  LX(25),LXI(25),LY(25),LYI(25),CXY(25),CXYI(25),CXY2(25),
     &CXY2I(25),CXY4(25),CXY4I(25),RLU(25),GC(25)
      REAL*4LA(100)
      REAL V(25,25,3)/1875*0.0/
      REAL*4ITSPEC(24)/96HARRAY SPECTRAL COEF. Y(I,J)=ALOG10(DA(I,J))
     &  R JOHNSON BOX 70                                              /
      REAL*8IARG(12)/96HFOURIER PHASE OF ITH RECEIVER
     &R. JOHNSON BOX 70                                               /
      REAL LARG(10)/40H0    1    2    34    5    6    78    9       /
      REAL LAF(10)/40H0    1    2    34    5    6    78    9       /
      REAL X(16)/0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0,11.0,12.0
     &,13.0,14.0,15.0/
      COMMON/TDT/T,DT,DT1,DT2,DT3
      COMMON/SIZE/NX,NY,NT,IIS1,IIS2
      COMMON/PGMC/IWRT,IDRW,IEN,IANL,ISAMP,IDFT,NWV,JDET,ITYPE
      COMMON/PARAM/R,LUO,LU,G,CO,C,DX,DY
      COMMON/STF/II   ,NARB,NARE
      COMMON/BARG/B(16),ARG(16),N1
      COMMON/TITLE/ITI,LA
```

```
      COMMON/SCALE/EX,EY,SLOPE,FACT
      COMMON/DRW/IU,IR,IXL,IYL,IW,IH,IG
      COMMON/EL/LX,LXI,LY,LYI,CXY,CXYI,CXY2,CXY2I,CXY4,CXY4I,RLU,GC
      DIMENSION IREAD(10),IWRITE(10)
      DIMENSION EA(40),EXF(40),FA(40),FX(40)
      DIMENSION Y(16)
C
C  NX=NO. OF INCREMENTS INTO WHICH THE X COORDINATE IS DIVIDED
C
C  NY=NO. OF INCREMENTS INTO WHICH THE Y COORDINATE IS DIVIDED
C
      TIME=ITIME(0)*0.01
      CALL CANCEL(2)
      READ(5,10)IREAD,IWRITE
      WRITE(6,20)NX,NY,NT,IIS1,IIS2,IREAD,IWRITE
      WRITE(6,260)T,DT1,DT2,DT3
      WRITE(6,170)R,G,DX,DY
      WRITE(6,180)IU,IR,IXL,IYL,IW,IH,IG
      WRITE(6,200)IWRT,IDRW,IEN,IANL,ISAMP,IDFT,NWV
      WRITE(6,210)EX,EY,SLOPE,FACT
      WRITE(6,240)II   ,NARB,NARE
      WRITE(6,30)DT1,NT
      READ(5,60)((ITI(I,J),J=1,12),I=1,10)
      WRITE(6,70)((ITI(I,J),J=1,12),I=1,10)
      READ(5,130)(LA(I),I=1,100)
      WRITE(6,140)(LA(I),I=1,100)
      READ(5,130)(EA(I),I=1,40),(FA(I),I=1,40),(EXF(I  ),I=1,40),(FX(I),I
     &=1,40)
      WRITE(6,220)(EA(I),I=1,40),(FA(I),I=1,40),(EXF(I),I=1,40),(FX(I),
     &I=1,40)
      CALL ICOND(V)
      WRITE(6,190)
      WRITE(6,160)(LX(I),LXI(I),LY(I),LYI(I),CXY(I),CXYI(I),I=1,NX)
      WRITE(6,280)
      WRITE(6,160)(CXY2(I),CXY2I(I),CXY4(I),CXY4I(I),RLU(I),GC(I),I=1,NX
     &)
      DT=DT1
      CALL GEN(V)
      ISAMP2=ISAMP/2
      DO 39 II=1,NT
      IIM1=II-1
      IF(IIM1.EQ.1)GO TO 1
      IF(MOD(IIM1,IWRT))3,1,3
    1 WRITE(6,40)T
      WTIME=ITIME(0)*0.01
      DO 2 I=1,NX
    2 WRITE(6,IWRITE)((I,J,K,V(I,J,K),K=1,NWV),J=1,NY)
      WTIME=ITIME(0)*0.01-WTIME
```

```
      WRITE(6,80)WTIME
    3 IF(IIM1.EQ.0)GO TO 5
      IF(II.EQ.NT)GO TO 4
      IF(MOD(IIM1,IDRW))5,4,5
    4 DTIME=ITIME(0)*0.01
      CALL DRAW3D(V)
      DTIME=ITIME(0)*0.01-DTIME
      WRITE(6,100)DTIME
    5 IF(IIM1.EQ.1)GO TO 6
      IF(MOD(IIM1,IEN))7,6,7
    6 MEB=MTIME(0)
      EN=ENERGY(V)
      MEE=MTIME(0)
      TE=(MEE-MEB)*26
      WRITE(6,50)EN
      WRITE(6,120)TE
      IF(IIM1.EQ.0)ENI=EN
      IF(ENI.EQ.0.0)ENI=1.0E-20
      PENL=(ENI-EN)*100.0/ENI
      WRITE(6,150)PENL
    7 WRITE(6,250)T
      MRKB=MTIME(0)
      CALL RKUTTA(V)
      MRKE=MTIME(0)
      TRK=(MRKE-MRKB)*26
      WRITE(6,290)TRK
      CALL STORE(V,DA)
      IF(MOD(II,IANL))9,8,9
    8 FTIME=ITIME(0)*0.01
      CALL FFT(DA,EA,EXF)
      FTIME=ITIME(0)*0.01-FTIME
      WRITE(6,230)FTIME
    9 CONTINUE
      IF(MOD(II,IDFT))19,11,19
   11 CALL DAMAX(DA,FMAX)
      MC=1
      DO 13 I=1,10
      DO 12 J=1,ISAMP2
   12 Y(J)=ALOG10(REAL(DA(I,J)))
      IF(I.EQ.10)MC=3
      CALL DRAW(ISAMP2,X,Y,MC,0,LAF(I),ITSPEC,2.,1.0,4,0,2,0,6,4,0,LAST)
      MC=2
   13 CONTINUE
      MC=1
      DO 15 I=1,10
      DO 14 J=1,ISAMP2
   14 Y(J)=AIMAG(DA(I,J))
      IF(I.EQ.10)MC=3
```

```
      CALL DRAW(ISAMP2,X,Y,MC,0,LARG(I),IARG,2., 90.,2,0,2,0,6,4,0,LAST)
      MC=2
   15 CONTINUE
   19 CONTINUE
      IF(II-IIS1)31,32,34
   31 DT=DT1
      T=T+DT
      GO TO 39
   32 DT=DT2
   33 T=T+DT
      GO TO 39
   34 IF(II-IIS2)33,35,36
   35 DT=DT3
   36 T=T+DT
   39 CONTINUE
      WRITE(6,40)T
      DO 41 I=1,NX
   41 WRITE(6,IWRITE)((I,J,K,V(I,J,K),K=1,NWV),J=1,NY)
      ETIME=ITIME(0)*0.01
      EN=ENERGY(V)
      ETIME=ITIME(0)*0.01-ETIME
      WRITE(6,50)EN
      PENL=(ENI-EN)*100.0/ENI
      WRITE(6,150)PENL
      WRITE(6,120)ETIME
      TIME=ITIME(0)*0.01-TIME
      WRITE(6,110)TIME
      STOP
   10 FORMAT(10A4,10A4)
   20 FORMAT(4H1NX=,I2,4H NY=,I2,4H NT=,I5,6H IIS1=,I3,6H IIS2=,I3//25H
     &INPUT READ FORMAT,IREAD=,10A4//28H OUTPUT WRITE FORMAT,IWRITE=,10A
     &4/)
   30 FORMAT(/' BEGIN INTEGRATION WITH STEP SIZE=',F10.6,' FOR NT=',I5,'
     & STEPS'/)
   40 FORMAT(/' ACCUSTIC PRESSURE AT POSITION I*J IN LATICE AT TIME T=',
     &F10.5//)
   50 FORMAT(/1X,'ENERGY STORED IN THE ARRAY AT ABOVE TIME =',1P1E14.7)
   60 FORMAT(6A8)
   70 FORMAT(/' GRAPH TITLES'//(1X,12A8))
   80 FORMAT(/' WRITE TIME=',F7.2,' SEC.')
   90 FORMAT( ' INTEGRATION TIME=',F7.2,' SEC.')
  100 FORMAT(/' GRAPH TIME=',F7.2,' SEC.')
  110 FORMAT(/' TOTAL PROGRAM TIME=',F7.2,' SEC.')
  120 FORMAT(' ENERGY TIME IN MICROSECONDS=',1P1E13.6)
  130 FORMAT(20A4)
  140 FORMAT(/' GRAPH LABEL'//(1X,20A4))
  150 FORMAT(/' PERCENTAGE ENERGY LOSS RELATIVE TO INITIAL ENERGY=',F12.
     &6)
```

```
160 FORMAT(//6(5X,1P1E13.6))
170 FORMAT(/3H R=,F10.5,3H G=,F10.5,4H DX=,F10.5,4H DY=,F10.5)
180 FORMAT(/4H IU=,I2,4H IR=,I2,5H IXL=,I2,5H IYL=,I2,4H IW=,I2,4H IH=
   &,I2,4H IG=,I2)
190 FORMAT(//5X,5HLX(I),13X,6HLXI(I),12X,5HLY(I),13X,6HLYI(I),12X,
   &6HCXY(I),12X,7HCXYI(I))
200 FORMAT(/6H IWRT=,I3,6H IDRW=,I3,5H IEN=,I3,6H IANL=,I3,7H ISAMP=,
   &I2,6H IDFT=,I3,5H IWV=,I1)
210 FORMAT(/4H EX=,F10.5,4H EY=,F10.5,7H SLOPE=,F10.5,6H FACT=,F10.5)
220 FORMAT(/14H WRITE FORMATS//8(1X,20A4/))
230 FORMAT(/10H FFT TIME=,F7.2,5H SEC.)
240 FORMAT(/4H II=  ,I4,6H NARB=,I2,6H NARE=,I2)
250 FORMAT(/32H PROBLEM TIME T, OF INTEGRATION=,F12.6,5H SEC.)
260 FORMAT(/3H T=,F12.5,5H DT1=,F12.6,5H DT2=,F12.6,5H DT3=,F12.6)
280 FORMAT(//5X,7HCXY2(I),6X,8HCXY2I(I),10X,7HCXY4(I),11X,8HCXY4I(I),
   &10X,6HRLU(I),12X,5HGC(I))
290 FORMAT(' INTEGRATION TIME IN MICROSECONDS=',1P1E14.7)
300 FORMAT(' ENERGY CALCULATION TIME IN MICROSECONDS=',1P1E14.7)
    END
```



```
    SUBROUTINE ICOND(V)
    REAL LUO
    REAL LU(25),C(25)
    REAL  LX(25),LXI(25),LY(25),LYI(25),CXY(25),CXYI(25),CXY2(25),
   &CXY2I(25),CXY4(25),CXY4I(25),RLU(25),GC(25)
    COMMON/PARAM/R,LUO,LU,G,CO,C,DX,DY
    COMMON/TDT/T,DT,DT1,DT2,DT3
    COMMON/SIZE/NX,NY,NT,IIS1,IIS2
    COMMON/EL/LX,LXI,LY,LYI,CXY,CXYI,CXY2,CXY2I,CXY4,CXY4I,RLU,GC
    COMMON/PGMC/IWRT,IDRW,IEN,IANL,ISAMP,IDFT,NWV,JDET,ITYPE
    DIMENSION V(25,25,3)
    DLU=0.
    DC=0.
    C(1)=CO
    LU(1)=LUO
    DXY=DX*DY
    DO 1 I=2,NX
    LU(I)=LU(I-1)-DLU
    C(I)=C(I-1)-DC
  1 CONTINUE
    DO 2 I=1,NX
    LX(I) =1./(LU(I)*DX)
    LXI(I)=0.5/LX(I)
    LY(I)=1./(LU(I)*DY)
    LYI(I)=0.5/LY(I)
```

```
      CXY(I)=1.0/(C(I)*DXY)
      CXYI(I)=0.5/CXY(I)
      CXY2(I)=2.0*CXY(I)
      CXY2I(I)=0.5/CXY2(I)
      CXY4(I)=4.0*CXY(I)
      CXY4I(I)=0.5/CXY4(I)
      GC(I)=G/C(I)
      RLU(I)=R/LU(I)
    2 CONTINUE
      RETURN
      END



      SUBROUTINE VEQUAS(V)
      COMMON/TDT/T,DT,DT1,DT2,DT3
      COMMON/SIZE/NX,NY,NT,IIS1,IIS2
      REAL   LX(25),LXI(25),LY(25),LYI(25),CXY(25),CXYI(25),CXY2(25),
     &CXY2I(25),CXY4(25),CXY4I(25),RLU(25),GC(25)
      COMMON/EL/LX,LXI,LY,LYI,CXY,CXYI,CXY2,CXY2I,CXY4,CXY4I,RLU,GC
      COMMON/PGMC/IWRT,IDRW,IEN,IANL,ISAMP,IDFT,NWV,JDET,ITYPE
      REAL DV(25,25,3)
      COMMON/DERIV/DV
      DIMENSION  V(25,25,3)
      NXM1=NX-1
      NYM1=NY-1
1111  DV(1,1,1)=-CXY4(1)*(V(1,1,2)+V(1,1,3))-GC(1)*V(1,1,1)
1112  DV(1,1,2)=LX(1)*(V(1,1,1)-V(1,2,1))-RLU(1)*V(1,1,2)
1113  DV(1,1,3)=LY(1)*(V(1,1,1)-V(2,1,1))-RLU(1)*V(1,1,3)
      DV(1,NY,1)=(LX(1)/CXY4(1))*DV(1,NYM1,2)
1911  DV(NX,1,1)=CXY4(NX)*(V(NXM1,1,3)-V(NX,1,2))-GC(NX)*V(NX,1,1)
1912  DV(NX,1,2)=LX(NX)*(V(NX,1,1)-V(NX,2,1))-RLU(NX)*V(NX,1,2)
1913  DV(NX,1,3)=0.
      DV(NX,NY,1)=(LX(NX)/CXY4(NX))*DV(NX,NYM1,2)
      IF(ITYPE)200,200,100
  100 DO 1000 I=2,NXM1
1011  DV(I,1,1)=CXY2(I)*(V(I-1,1,3)-V(I,1,3)-V(I,1,2))-GC(I)*V(I,1,1)
1012  DV(I,1,2)=LX(I)*(V(I,1,1)-V(I,2,1))-RLU(I)*V(I,1,2)
1013  DV(I,1,3)=LY(I)*(V(I,1,1)-V(I+1,1,1))-RLU(I)*V(I,1,3)
      DV(I,NY,1)=(LX(I)/CXY2(I))*DV(I,NYM1,2)
      DO 1000 J=2,NYM1
    1 DV(I,J,1)=CXY(I)*(V(I,J-1,2)-V(I,J,2)+V(I-1,J,3)-V(I,J,3))-GC(I)*
     &V(I,J,1)
    2 DV(I,J,2)= LX(I)*(V(I,J,1)-V(I,J+1,1))-RLU(I)*V(I,J,2)
    3 DV(I,J,3)= LY(I)*(V(I,J,1)-V(I+1,J,1))-RLU(I)*V(I,J,3)
 1000 CONTINUE
      DO 2000 J=2,NYM1
```

```
1101 DV(1,J,1)=CXY2(I)*(V(1,J-1,2)-V(1,J,2)-V(1,J,3))-GC(1)*V(1,J,1)
1102 DV(1,J,2)=LX(1)*(V(1,J,1)-V(1,J+1,1))-RLU(1)*V(1,J,2)
1103 DV(1,J,3)=LY(1)*(V(1,J,1)-V(2,J,1))-RLU(1)*V(1,J,3)
1901 DV(NX,J,1)=CXY2(I)*(V(NX,J-1,2)-V(NX,J,2)+V(NXM1,J,3))-GC(NX)*V(NX
     &,J,1)
1902 DV(NX,J,2)=LX(NX)*(V(NX,J,1)-V(NX,J+1,1))-RLU(NX)*V(NX,J,2)
1903 DV(NX,J,3)=0.
2000 CONTINUE
     RETURN
 200 DO 3000 I=2,NXM1
     DV(I,1,2)=LX(1)*(V(I,1,1)-V(I,2,1))-RLU(1)*V(I,1,2)
     DV(I,1,3)=LY(1)*(V(I,1,1)-V(I+1,1,1))-RLU(1)*V(I,1,3)
     DV(I,NY,2)=0.
     DV(I,NY,3)=LY(1)*(V(I,NY,1)-V(I+1,NY,1))-RLU(1)*V(I,NY,3)
     DO 3000 J=2,NYM1
     DV(I,J,2)= LX(1)*(V(I,J,1)-V(I,J+1,1))-RLU(1)*V(I,J,2)
     DV(I,J,3)= LY(1)*(V(I,J,1)-V(I+1,J,1))-RLU(1)*V(I,J,3)
3000 CONTINUE
     DO 4000 J=2,NYM1
     DV(1,J,2)=LX(1)*(V(1,J,1)-V(1,J+1,1))-RLU(1)*V(1,J,2)
     DV(1,J,3)=LY(1)*(V(1,J,1)-V(2,J,1))-RLU(1)*V(1,J,3)
     DV(NX,J,2)=LX(1 )*(V(NX,J,1)-V(NX,J+1,1))-RLU(1 )*V(NX,J,2)
     DV(NX,J,3)=0.
4000 CONTINUE
     RETURN
     END



     SUBROUTINE RKUTTA(V)
     REAL CRK(4)/0.0,0.5,0.5,1.0/
     REAL F16TH/0.1666667/
     COMMON/TDT/T,DT,DT1,DT2,DT3
     COMMON/SIZE/NX,NY,NT,IIS1,IIS2
     COMMON/PGMC/IWRT,IDRW,IEN,IANL,ISAMP,IDFT,NWV,JDET,ITYPE
     COMMON/STF/II   ,NARB,NARE
     REAL DV(25,25,3)
     COMMON/DERIV/DV
     DIMENSION V(25,25,3),VC(25,25,3),AK(25,25,3,4)
     CALL VEQUAS(V)
     DO 1 K=1,3
     DO 1 J=1,NY
     DO 1 I=1,NX
   1 AK(I,J,K,1)=DT*DV(I,J,K)
     DO 3 L=2,4
     LM1=L-1
     CRKD=CRK(L)
```

```
      DO 2 K=1,3
      DO 2 J=1,NY
      DO 2 I=1,NX
    2 VC(I,J,K)=V(I,J,K)+CRKD*AK(I,J,K,LM1)
      CALL VEQUAS(VC)
      DO 3 K=1,3
      DO 3 J=1,NY
      DO 3 I=1,NX
    3 AK(I,J,K,L)=DT*DV(I,J,K)
      DO 4 K=1,3
      DO 4 J=1,NY
      DO 4 I=1,NX
    4 V(I,J,K)=V(I,J,K)+(AK(I,J,K,1)+2.*(AK(I,J,K,2)+AK(I,J,K,3))+AK(I,J
     &,K,4))*F16TH
      RETURN
      END



      FUNCTION ENERGY(V)
      REAL   LX(25),LXI(25),LY(25),LYI(25),CXY(25),CXYI(25),CXY2(25),
     &CXY2I(25),CXY4(25),CXY4I(25),RLU(25),GC(25)
      COMMON/EL/LX,LXI,LY,LYI,CXY,CXYI,CXY2,CXY2I,CXY4,CXY4I,RLU,GC
      COMMON/TDT/T,DT,DT1,DT2,DT3
      COMMON/SIZE/NX,NY,NT,IIS1,IIS2
      COMMON/PGMC/IWRT,IDRW,IEN,IANL,ISAMP,IDFT,NWV,JDET,ITYPE
      DIMENSION V(25,25,3)
      NXM1=NX-1
      NYM1=NY-1
      EN=0.
      DO 1000 I=2,NXM1
1011  EN=EN+CXY2I(I)*V(I,1,1)**2
1012  EN=EN+LXI(I)*V(I,1,2)**2
1013  EN=EN+LYI(I)*V(I,1,3)**2
1091  EN=EN+CXY2I(I)*V(I,NY,1)**2
1092  EN=EN
1093  EN=EN+LYI(I)*V(I,NY,3)**2
      DO 1000 J=2,NYM1
    1 EN=EN+CXYI(I)*V(I,J,1)**2
    2 EN=EN + LXI(I)*V(I,J,2)**2
    3 EN=EN+LYI(I)*V(I,J,3)**2
1000  CONTINUE
      DO 2000 J=2,NYM1
1101  EN=EN+CXY2I(1)*V(1,J,1)**2
1102  EN=EN+LXI(1)*V(1,J,2)**2
1103  EN=EN+LYI(1)*V(1,J,3)**2
1901  EN=EN+CXY2I(NX)*V(NX,J,1)**2
```

```
 1902 EN=EN+LXI(NX)*V(NX,J,2)**2
 1903 EN=EN
 2000 CONTINUE
 1111 EN=EN+CXY4I(1)*V(1,1,1)**2
 1112 EN=EN+LXI(1)*V(1,1,2)**2
 1113 EN=EN+LYI(1)*V(1,1,3)**2
 1191 EN=EN+CXY4I(1)*V(1,NY,1)**2
 1192 EN=EN
 1193 EN=EN+LYI(1)*V(1,NY,3)**2
 1911 EN=EN+CXY4I(NX)*V(NX,1,1)**2
 1912 EN=EN+LXI(NX)*V(NX,1,2)**2
 1913 EN=EN
 1991 EN=EN+CXY4I(NX)*V(NX,NY,1)**2
 1992 EN=EN
 1993 EN=EN
      ENERGY=EN
      RETURN
      END



      SUBROUTINE DRAW3D(V)
C DRAW3D USES SUBROUTINE DRAW TO MAKE A CONTOUR MAP SHOWING THE PRESSURE
C DISTRIBUTION FOR TWO-DIMENSIONAL PROBLEMS. FOR THREE-DIMENSIONAL PROBLEMS
C TWO DIMENSIONAL LAYER PLOTS MUST BE USED.
      REAL*8ITI(10,12),IT(12)
      REAL*4LA(100)
      INTEGER NG/1/
      COMMON/TDT/T,DT,DT1,DT2,DT3
      COMMON/SIZE/NX,NY,NT,IIS1,IIS2
      COMMON/PGMC/IWRT,IDRW,IEN,IANL,ISAMP,IDFT,NWV,JDET,ITYPE
      COMMON/SCALE/EX,EY,SLOPE,FACT
      COMMON/TITLE/ITI,LA
      COMMON/DRW/IU,IR,IXL,IYL,IW,IH,IG
      DIMENSION V(25,25,3),X(25),Y(25),Z(25),TEMP(25,25)
      MC=1
      DO 1 I=1,12
    1 IT(I)=ITI(NG,I)
      VMAX=0.0
      DO 11 J=1,NY
      DO 11 I=1,NX
      TEMP(I,J)=V(I,J,1)
      VABS=ABS(TEMP(I,J))
      IF(VABS.GT.VMAX)VMAX=VABS
   11 CONTINUE
      IF(VMAX.NE.1.0)GO TO 12
      GO TO 14
```

```
   12 DEL=1.0/VMAX
      DO 13 J=1,NY
      DO 13 I=1,NX
   13 TEMP(I,J)=DEL*TEMP(I,J)
      WRITE(6,20)DEL
   14 DO 3 I=1,NX
      FIM1=I-1
      DO 2 M=1,NY
      FMM1=M-1
      Z(M)=FACT*TEMP(I,M)
      X(M)=FMM1+FIM1*SLOPE
    2 Y(M)=FIM1+Z(M)
      IF(I.GT.1)MC=2
    3 CALL DRAW(NY,X,Y,MC,0,LA(I),IT,EX,EY,IU,IR,IXL,IYL,IW,IH,IG,LAST)
      DO 5 J=1,NY
      FJM1=J-1
      DO 4 M=1,NX
      FMM1=M-1
      Z(M)=FACT*TEMP(M,J)
      X(M)=FJM1+FMM1*SLOPE
    4 Y(M)=Z(M)+FMM1
      IF(J.EQ.NY)MC=3
    5 CALL DRAW(NX,X,Y,MC,0,LA(J),IT,EX,EY,IU,IR,IXL,IYL,IW,IH,IG,LAST)
      NG=NG+1
   20 FORMAT(///5H DEL=,F10.4///)
      RETURN
      END



      SUBROUTINE GEN(V)
C GEN IS THE SOURCE SUBROUTINE
      COMMON/TDT/T,DT,DT1,DT2,DT3
      COMMON/SIZE/NX,NY,NT,IIS1,IIS2
      COMMON/PGMC/IWRT,IDRW,IEN,IANL,ISAMP,IDFT,NWV
      REAL PI2/6.28318/,F/  4.0/
      DIMENSION V(25,25,3)
      FUN=75.0
      PI2T=6.283185*T
      V( 2 , 2 ,1)=1.0*COS(PI2T*FUN)
      RETURN
      END
```

```
      SUBROUTINE STORE(V,DA)
      COMPLEX*8DA(10,16)
      COMMON/SIZE/NX,NY,NT,IIS1,IIS2
      COMMON/STF/II   ,NARB,NARE
      COMMON/PGMC/IWRT,IDRW,IEN,IANL,ISAMP,IDFT,NWV,JDET,ITYPE
      DIMENSION V(25,25,3)
      INDEX=MCD(II,ISAMP)
      IF(INDEX.EQ.0)INDEX=ISAMP
      DO 1 I=1,10
    1 DA(I,INDEX)=V(I,JDET,1)
      RETURN
      END



      SUBROUTINE CALG1(A,FORM)
      COMPLEX*8A(16)
      COMMON/BARG/B(16),ARG(16),N1
      COMMON/PGMC/IWRT,IDRW,IEN,IANL,ISAMP,IDFT,NWV,JDET,ITYPE
      DIMENSION IM1(16)
      DIMENSION FORM(40)
      DO 1 I1=1,ISAMP
      IM1(I1)=I1-1
      B(I1)=CABS(A(I1))
      AR=REAL(A(I1))
      AI=AIMAG(A(I1))
      IF(AR.EQ.0.0)AR=1.0E-06
    1 ARG(I1)=57.29578*ATAN2(AI,AR)
      WRITE(6,FORM)(IM1(I1),A(I1),B(I1),ARG(I1),I1=1,ISAMP)
      RETURN
      END



      SUBROUTINE FFT(DA,FORMD,FORMF)
      COMPLEX*8DA(10,16),PI2J/(0.0,6.28318)/
      COMPLEX*8DU(16)
      INTEGER M(3)/4,0,0/
      COMMON/BARG/B(16),ARG(16),N1
      COMMON/PGMC/IWRT,IDRW,IEN,IANL,ISAMP,IDFT,NWV,JDET,ITYPE
      COMMON/STF/II   ,NARB,NARE
      DIMENSION FORMD(40),FORMF(40)
      DIMENSION INV(16),S(16)
      DO 2 I=1,10
      DO 1 J=1,ISAMP
    1 DU(J)=DA(I,J)
```

```
      CALL CALG1(DU,FORMD)
      CALL HARM(DU,M,INV,S,-1,IFERR)
      CALL CALG1(DU,FORMF)
      DO 2 J=1,ISAMP
    2 DA(I,J)=B(J)+(0.0,1.0)*ARG(J)
      RETURN
      END



      SUBROUTINE DAMAX(DA,FMAX)
      COMPLEX DA(10,16)
      COMMON/PGMC/IWRT,IDRW,IEN,IANL,ISAMP,IDFT,NWV,JDET,ITYPE
      FMAX=0.0
      DO 1 I=1,10
      DO 1 J=1,ISAMP
      TEMP=REAL(DA(I,J))
      IF(TEMP.GT.FMAX)FMAX=TEMP
    1 CONTINUE
      FMAX=FMAX/15.
      RETURN
      END
C GENERAL 3-DIMENSIONAL OVERLAY PROGRAM FOR INTEGRATION OF THE WAVE EQUATION
C USING RUNGE-KUTTA 4-TH ORDER INTEGRATION OF THE CONTINUOUS DISCRETE SYSTEM
C THE PRESSURE AND 3 VELOCITY COMPONENTS ARE STORED IN THE LARGE ARRAY
C V(I,J,KLM) WITH THE TIME DERIVATIVE IN DV(IJKL). ALL CALCULATIONS ARE ACTUALLY
C PERFORMED ON THE LARGE ARRAY V(I,J,K,L,M) WHICH IS INDEXED AS A SINGLE
C SUBSCRIPT ARRAY WITH INDEX IJKLM. OVERLAY ARRAYS OF BOUNDARY VALUES ARE STORED
C ON MAGNETIC DISC FILE 8 WITH 15000000 60 BIT WORDS WHEN RUN ON A CDC6500.
C ANY OF THE OTHER COMMON INTEGRATION SCHEMES THAT EVALUATE THE TIME DERIVATIVE
C DV(I)/DT AS A FUNCTION OF THE PRESENT VALUES OF THE VARIABLES V(I) MAY BE USED
C IN PLACE OF RKUT SUCH AS RUNGE-KUTTA-ADAMS-MOULTON,ADAMS-BASHFORTH,ETC.
      DIMENSION V(10,5,360),DV(4500),A(16),AMP(4),S(4),INV(4),M(3),      WV3D0000
     *BNEW(8,600),CNEW(8,600),BOLD(8,600),COLD(8,600),IIN(4)             WV3D0010
      INTEGER ARRAYS,OK                                                  WV3D0020
      COMMON/MEDIUM/VMD(400),VMDV(400),BMD(400),BMDI(400),FX(400),       WV3D0030
     *FY(400),FZ(400),HX(400),HY(400),HZ(400),ZSIDE(400),YSIDE(400),     WV3D0040
     *ZFRT(100),ZBACK(100),YBACK(100),ZTOP(100),YTOP(100),ZBOT(100)      WV3D0050
      COMMON/SIZE/N1M1,N1,N1P1,N2M1,N2,N2P1,N3M1,N3,N3P1,N12,N13,N14,N15WV3D0060
     *,N23,I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,I11,I12,I13,I14,I15,I16,I17,LTWV3D0070
     *,N132,N133,N142,N143                                               WV3D0080
      COMMON/SIG/NXYZS,NFREQ                                             WV3D0090
      COMMON/TDT/T,DT(20),TS,DX,DY,DZ                                    WV3D0100
      EQUIVALENCE(NX,N1),(NY,N2),(NZ,N3)                                 WV3D0110
      DATA NW1/10/,NW2/5/,NW3/10/,M(1)/3/,M(2)/0/,M(3)/0/,IWRT/50/       WV3D0120
      DATA IEN/1/,II/0/,KK/0/,NSAMP/8/,IIMIN/60 /,INDEX/0/,IDTS/20/      WV3D0130
      DATA ICW/1/,IBW/1/,IIN(1)/0/,IIN(2)/1/,IIN(3)/2/,IIN(4)/3/,IPW/1/  WV3D0140
```

```
      DATA ISHS/1/,ISHB/1/,ISHBK/1/,ISHF/1/,ISHT/1/,IS/1/,IT/0/,IB/0/   WV3DC150
      DATA RANGE/0.0/,RANGEM/100.0/,STABLE/10.0/,ERROR/0.01/,UAVE/0.0/  WV3D0160
      DATA DEPTH/0.0/,DEPTHM/100.0/                                     WV3DC170
C                                                                       WV3DC180
CC V(IJKLM)=V(I+(J-1)*N1+(K-1)*N12+(L-1)*N13+(M-1)*N14)                 WV3D0190
C                                                                       WV3DC200
      READ(5,10)NX,NY,NZ,NT,NXS,NYS,NZS,NFREQ                           WV3DC210
      WRITE(6,290)                                                      WV3DC220
      WRITE(6,40)NX,NY,NZ,NT,NXS,NYS,NZS,NFREQ                          WV3DC230
      READ(5,150)DX,DY,DZ                                               WV3DC240
      READ(5, 20)(DT(I),I=1,20)                                         WV3DC250
      WRITE(6,160)DX,DY,DZ                                              WV3D0260
      WRITE(6,360)(DT(I),I=1,20)                                        WV3DC270
      READ(5,150)FUN,(AMP(I),I=1,NFREQ)                                 WV3D0280
      WRITE(6,180)FUN,(AMP(I),I=1,NFREQ)                                WV3DC290
      N1M1=N1-1                                                         WV3D0300
      N2M1=N2-1                                                         WV3D0310
      N3M1=N3-1                                                         WV3D0320
      N1P1=N1+1                                                         WV3D0330
      N2P1=N2+1                                                         WV3D0340
      N3P1=N3+1                                                         WV3D0350
      N12=N1*N2                                                         WV3D0360
      N13=N12*N3                                                        WV3D0370
      N14=N13*4                                                         WV3D0380
      N15=N14*6                                                         WV3D0390
      N132=2*N13                                                        WV3D0400
      N133=3*N13                                                        WV3D0410
      N142=2*N14                                                        WV3D0420
      N23=N2*N3                                                         WV3D0430
      N24=N23*4                                                         WV3D0440
      I1=-N1-N12                                                        WV3D0450
      I2 =-N13-N12-N1                                                   WV3D0460
      I3=1-N12-N1                                                       WV3D0470
      I4=1-N1-N12+N13                                                   WV3D0480
      I5=N1M1-N1-N12+N13                                                WV3D0490
      I6=-N2-N23                                                        WV3D0500
      I7=-N12+N132                                                      WV3D0510
      I8=-N1+N12-N13                                                    WV3D0520
      I9=2-N1-N12-N13                                                   WV3D0530
      I10=-N1+N132                                                      WV3D0540
      I11=N12-N1                                                        WV3D0550
      I12=N13-N12                                                       WV3D0560
      I13=-N12+N133                                                     WV3D0570
      N1N3=N1*N3                                                        WV3D0580
      KLMT=24*N3                                                        WV3D0590
      NXYZS=NXS+(NYS-1 )*N1 +(NZS-1)*N12                                WV3D0600
      T=0.0                                                             WV3D0610
      WRITE(6, 60)N1,N2,N3,      N12,N13,N14,N15                        WV3D0620
```

```
      ARRAYS=N15+N14+12*N1*N3+6*N1*N2                                    WV3D0630
      WRITE(6,90)ARRAYS                                                  WV3D0640
      VOL2=0.5*DX*DY*DZ                                                  WV3D0650
      DO 1 K=1,N3                                                        WV3D0660
      IK=-N1+K*N1                                                        WV3D0670
      KI=IK                                                              WV3D0680
      DO 1 I=1,N1                                                        WV3D0690
      IK=KI+I                                                            WV3D0700
      VMD(IK)=1.0E+03                                                    WV3D0710
      VMDV(IK)=VOL2*VMD(IK)                                              WV3D0720
      BMD(IK)=2.25E+09                                                   WV3D0730
      BMDI(IK)=VOL2/BMD(IK)                                              WV3D0740
      HX(IK)=1.0/(DX*VMD(IK))                                            WV3D0750
      HY(IK)=1.0/(DY*VMD(IK))                                            WV3D0760
      HZ(IK)=1.0/(DZ*VMD(IK))                                            WV3D0770
      FX(IK)=BMD(IK)/DX                                                  WV3D0780
      FY(IK)=BMD(IK)/DY                                                  WV3D0790
      FZ(IK)=BMD(IK)/DZ                                                  WV3D0800
      ZSIDE(IK)=SQRT(VMD(IK)*BMD(IK))                                    WV3D0810
      IF(ISHS.EQ.0)ZSIDE(IK)=0.0                                         WV3D0820
      YSIDE(IK)=HY(IK)                                                   WV3D0830
    1 CONTINUE                                                           WV3D0840
      DO 2 KLM=1,KLMT                                                    WV3D0850
      DO 2 J=1,N2                                                        WV3D0860
      DO 2 I=1,N1                                                        WV3D0870
    2 V(I,J,KLM)=0.0                                                     WV3D0880
      DO 3 IJKL=1,N14                                                    WV3D0890
    3 DV(IJKL)=0.0                                                       WV3D0900
      DO 5 K=1,N3                                                        WV3D0910
      JK=-N2+K*N2                                                        WV3D0920
      N1M1K=-1+K*N1                                                      WV3D0930
      OK=1-N1+K*N1                                                       WV3D0940
      KJ=JK                                                              WV3D0950
      DO 5 J=1,N2                                                        WV3D0960
      JK=KJ+J                                                            WV3D0970
      ZFRT(JK)=ZSIDE(N1M1K)                                              WV3D0980
      IF(ISHF.EQ.0)ZFRT(JK)=0.0                                          WV3D0990
      ZBACK(JK)=ZSIDE(OK)                                                WV3D1000
      IF(ISHBK.EQ.0)ZBACK(JK)=0.0                                        WV3D1010
      YBACK(JK)=HX(OK)                                                   WV3D1020
    5 CONTINUE                                                           WV3D1030
      DO 6 J=1,N2                                                        WV3D1040
      IJ=-N1+J*N1                                                        WV3D1050
      JI=IJ                                                              WV3D1060
      DO 6 I=1,N1                                                        WV3D1070
      IJ=JI+I                                                            WV3D1080
      INM=N1N3-2*N1+I                                                    WV3D1090
      ZTOP(IJ)=ZSIDE(I)                                                  WV3D1100
```

```
      IF(ISHT.EQ.0)ZTOP(IJ)=0.0                                         WV3D1110
      YTOP(IJ)=HZ(I)                                                    WV3D1120
      ZBOT(IJ)=ZSIDE(INM)                                               WV3D1130
      IF(ISHB.EQ.0)ZBOT(IJ)=0.0                                         WV3D1140
    6 CONTINUE                                                          WV3D1150
      IF(IPW.EQ.0)GO TO 8                                               WV3D1160
      WRITE(6,130)                                                      WV3D1170
      WRITE(6,170)(VMD (IK),IK=1,N1N3)                                  WV3D1180
      WRITE(6,70)                                                       WV3D1190
      WRITE(6,170)(VMDV(IK),IK=1,N1N3)                                  WV3D1200
      WRITE(6,140)                                                      WV3D1210
      WRITE(6,170)(BMD (IK),IK=1,N1N3)                                  WV3D1220
      WRITE(6,80)                                                       WV3D1230
      WRITE(6,170)(BMDI(IK),IK=1,N1N3)                                  WV3D1240
      WRITE(6,340)                                                      WV3D1250
      WRITE(6,170)(ZSIDE(IK),IK=1,N1N3)                                 WV3D1260
      WRITE(6,350)                                                      WV3D1270
      WRITE(6,170)(YSIDE(IK),IK=1,N1N3)                                 WV3D1280
      WRITE(6,210)                                                      WV3D1290
      WRITE(6,170)(HX(IK),IK=1,N1N3)                                    WV3D1300
      WRITE(6,220)                                                      WV3D1310
      WRITE(6,170)(HY(IK),IK=1,N1N3)                                    WV3D1320
      WRITE(6,230)                                                      WV3D1330
      WRITE(6,170)(HZ(IK),IK=1,N1N3)                                    WV3D1340
      WRITE(6,240)                                                      WV3D1350
      WRITE(6,170)(FX(IK),IK=1,N1N3)                                    WV3D1360
      WRITE(6,250)                                                      WV3D1370
      WRITE(6,170)(FY(IK),IK=1,N1N3)                                    WV3D1380
      WRITE(6,260)                                                      WV3D1390
      WRITE(6,170)(FZ(IK),IK=1,N1N3)                                    WV3D1400
      WRITE(6,270)                                                      WV3D1410
      WRITE(6,170)(ZFRT(JK),JK=1,N23)                                   WV3D1420
      WRITE(6,280)                                                      WV3D1430
      WRITE(6,170)(ZBACK(JK),JK=1,N23)                                  WV3D1440
      WRITE(6,300)                                                      WV3D1450
      WRITE(6,170)(YBACK(JK),JK=1,N23)                                  WV3D1460
      WRITE(6,320)                                                      WV3D1470
      WRITE(6,170)(ZTOP (IJ),IJ=1,N12)                                  WV3D1480
      WRITE(6,330)                                                      WV3D1490
      WRITE(6,170)(YTOP (IJ),IJ=1,N12)                                  WV3D1500
      WRITE(6,310)                                                      WV3D1510
      WRITE(6,170)(ZBOT (IJ),IJ=1,N12)                                  WV3D1520
    8 CONTINUE                                                          WV3D1530
      I7=1                                                              WV3D1540
      IX=1                                                              WV3D1550
    9 CONTINUE                                                          WV3D1560
      II=II+1                                                           WV3D1570
      IIM1=II-1                                                         WV3D1580
```

```
      IF(II.LT.21)LT=II                                                 WV3D1590
      T=T+DT(LT)                                                        WV3D1600
      IF(IS.EQ.1)CALL SOURCE(V,AMP,FUN)                                 WV3D1610
      IF(IB.EQ.1)CALL BACK(V,BOLD,FUN)                                  WV3D1620
      IF(IT.EQ.1)CALL TOP(V,COLD,FUN)                                   WV3D1630
      CALL RKUT(V,DV)                                                   WV3D1640
C AVERAGE ENERGY CALCULATION
      ENERGY=U(V)                                                       WV3D1650
      UAVE=UAVE+(ENERGY-UAVE)*DT(LT)/T                                  WV3D1660
      IF(II.GT.IIMIN)STABLE=ABS(UAVE-SUAVE)                             WV3D1670
      ERROR=0.1*UAVE                                                    WV3D1680
      SUAVE=UAVE                                                        WV3D1690
C     IF(MOD(II,IPLOT).EQ.0)CALL DRAW3D(V)                              WV3D1700
      IF(MOD(IIM1,IWRT))14,13,14                                        WV3D1710
   13 WRITE(6,400)T                                                     WV3D1720
      WRITE(6,100)(((I,J,KLM,V(I,J,KLM),KLM=1,NW3),J=1,NW2),I=1,NW1)    WV3D1730
   14 IF(MOD(IIM1,IEN))16,15,16                                         WV3D1740
   15 WRITE(6,50)ENERGY,II,T,UAVE,STABLE                                WV3D1750
      IF(IIM1.EQ.0)GO TO 16                                             WV3D1760
      PEC=(ENERGY-ELAST)*100.0/ELAST                                    WV3D1770
      WRITE(6,110)PEC                                                   WV3D1780
   16 CONTINUE                                                          WV3D1790
      ELAST=ENERGY                                                      WV3D1800
      IF(STABLE.LT.ERROR)GO TO 17                                       WV3D1810
      IISS=II+1                                                         WV3D1820
      TSS=T+DT(LT)                                                      WV3D1830
      IF(II.GT.NT)STOP                                                  WV3D1840
      GO TO 9                                                           WV3D1850
   17 IF(II.EQ.IISS)GO TO 18                                            WV3D1860
      IF(II-IINS)9,18,18                                                WV3D1870
   18 IINS=II+IDTS                                                      WV3D1880
      INDEX=INDEX+1                                                     WV3D1890
      DO 19 L=1,4                                                       WV3D1900
      JKLE=I6+L*N23                                                     WV3D1910
      LK=JKLE                                                           WV3D1920
      KL=-N3+L*N3                                                       WV3D1930
      LK1=KL                                                            WV3D1940
      DO 19 K=2,N3M1                                                    WV3D1950
      JKLE=LK+K*N2                                                      WV3D1960
      KJ=JKLE                                                           WV3D1970
      KL=LK1+K                                                          WV3D1980
      DO 19 J=2,N2M1                                                    WV3D1990
      JKLE=2*(KJ+J)                                                     WV3D2000
      JKLO=JKLE-1                                                       WV3D2010
      BNEW(INDEX,JKLE)=0.0                                              WV3D2020
   19 BNEW(INDEX,JKLO)=V(N1M1,J,KL)                                     WV3D2030
      DO 21 L=1,4                                                       WV3D2040
      IJLE=I1+L*N12                                                     WV3D2050
```

```
      LJ=IJLE                                                           WV3D2060
      NML=-1+L*N3                                                       WV3D2070
      DO 21 J=2,N2M1                                                    WV3D2080
      IJLE=LJ+J*N1                                                      WV3D2090
      JI=IJLE                                                           WV3D2100
      DO 21 I=2,N1M1                                                    WV3D2110
      IJLE=2*(JI+I)                                                     WV3D2120
      IJLO=IJLE-1                                                       WV3D2130
      CNEW(INDEX,IJLE)=C.0                                              WV3D2140
   21 CNEW(INDEX,IJLO)=V(I,J,NML)                                       WV3D2150
      WRITE(6,10)INDEX,II                                               WV3D2160
      IF(INDEX-NSAMP)9,23,23                                            WV3D2170
   23 IF(ICW.EQ.1)WRITE(6,370)IZ,IX                                     WV3D2180
      DO 27 L=1,4                                                       WV3D2190
      IJLE=II+L*N12                                                     WV3D2200
      LJ=IJLE                                                           WV3D2210
      DO 27 J=2,N2M1                                                    WV3D2220
      IJLE=LJ+J*N1                                                      WV3D2230
      JI=IJLE                                                           WV3D2240
      DO 27 I=2,N1M1                                                    WV3D2250
      IJLE=2*(JI+I)                                                     WV3D2260
      IJLO=IJLE-1                                                       WV3D2270
      DO 24 INDEX=1,NSAMP                                               WV3D2280
      IND2=2*INDEX                                                      WV3D2290
      IND2M=IND2-1                                                      WV3D2300
      A(IND2)=C.0                                                       WV3D2310
   24 A(IND2M)=CNEW(INDEX,IJLO)                                         WV3D2320
      CALL HARM(A,M,INV,S,-1,IFERR)                                     WV3D2330
      DO 25 INDEX=1,NFREQ                                               WV3D2340
      IND2=2*INDEX                                                      WV3D2350
      IND2M=IND2-1                                                      WV3D2360
      CNEW(INDEX,IJLE)=ATAN2(A(IND2),A(IND2M))                          WV3D2370
   25 CNEW(INDEX,IJLO)=2.0*SQRT(A(IND2)**2+A(IND2M)**2)                 WV3D2380
      WRITE(8)(CNEW(IN,IJLO),CNEW(IN,IJLE),IN=2,NFREQ)                  WV3D2390
      IF(ICW.EQ.1)WRITE(6,200)(IIN(IN),I,J,N3M1,L,CNEW(IN,IJLO),CNEW(IN,WV3D2400
     *IJLE),IN=1,NFREQ)                                                 WV3D2410
   27 CONTINUE                                                          WV3D2420
      IF(IBW.EQ.1)WRITE(6,380)IZ,IX                                     WV3D2430
      DO 34 L=1,4                                                       WV3D2440
      JKLE=I6+L*N23                                                     WV3D2450
      LK=JKLE                                                           WV3D2460
      DO 34 K=2,N3M1                                                    WV3D2470
      JKLE=LK+K*N2                                                      WV3D2480
      KJ=JKLE                                                           WV3D2490
      DO 34 J=2,N2M1                                                    WV3D2500
      JKLE=2*(KJ+J)                                                     WV3D2510
      JKLO=JKLE-1                                                       WV3D2520
      DO 31 INDEX=1,NSAMP                                               WV3D2530
```

```
      IND2=2*INDEX                                                      WV3D2540
      IND2M=IND2-1                                                      WV3D2550
      A(IND2)=0.0                                                       WV3D2560
   31 A(IND2M)=BNEW(INDEX,JKLO)                                         WV3D2570
      CALL HARM(A,M,INV,S,-1,IFERR)                                     WV3D2580
      DO 32 INDEX=1,NFREQ                                               WV3D2590
      IND2=2*INDEX                                                      WV3D2600
      IND2M=IND2-1                                                      WV3D2610
      BNEW(INDEX,JKLE)=ATAN2(A(IND2),A(IND2M))                          WV3D2620
   32 BNEW(INDEX,JKLO)=2.0*SQRT(A(IND2)**2+A(IND2M)**2)                 WV3D2630
      WRITE(8)(BNEW(IN,JKLO),BNEW(IN,JKLE),IN=2,NFREQ)                  WV3D2640
      IF(IBW.EQ.1)WRITE(6,200)(IIN(IN),N1M1,J,K,L,BNEW(IN,JKLO),BNEW(IN,WV3D2650
     *JKLE),IN=1,NFREQ)                                                 WV3D2660
   34 CONTINUE                                                          WV3D2670
      REWIND 8                                                          WV3D2680
      DO 41 KLM=1,KLMT                                                  WV3D2690
      DO 41 J=1,N2                                                      WV3D2700
      DO 41 I=1,N1                                                      WV3D2710
   41 V(I,J,KLM)=0.0                                                    WV3D2720
      DO 42 IJKL=1,N14                                                  WV3D2730
   42 DV(IJKL)=0.0                                                      WV3D2740
      IF(IX.EQ.1.AND.IZ.EQ.1)GO TO 51                                   WV3D2750
      IF(IZ.EQ.1)GO TO 53                                               WV3D2760
   51 CONTINUE                                                          WV3D2770
      DO 52 L=1,4                                                       WV3D2780
      IJLE=I1+L*N12                                                     WV3D2790
      LJ=IJLE                                                           WV3D2800
      DO 52 J=2,N2M1                                                    WV3D2810
      IJLE=LJ+J*N1                                                      WV3D2820
      JI=IJLE                                                           WV3D2830
      DO 52 I=2,N1M1                                                    WV3D2840
      IJLE=2*(JI+I)                                                     WV3D2850
      IJLO=IJLE-1                                                       WV3D2860
      READ(8)(COLD(IN,IJLO),COLD(IN,IJLE),IN=2,NFREQ)                   WV3D2870
   52 CONTINUE                                                          WV3D2880
   53 CONTINUE                                                          WV3D2890
      IF(IX.EQ.1)GO TO 69                                               WV3D2900
      DO 62 L=1,4                                                       WV3D2910
      JKLE=I6+L*N23                                                     WV3D2920
      LK=JKLE                                                           WV3D2930
      DO 62 K=2,N3M1                                                    WV3D2940
      JKLE=LK+K*N2                                                      WV3D2950
      KJ=JKLE                                                           WV3D2960
      DO 62 J=2,N2M1                                                    WV3D2970
      JKLE=2*(KJ+J)                                                     WV3D2980
      JKLO=JKLE-1                                                       WV3D2990
      READ(8)(BOLD(IN,JKLO),BOLD(IN,JKLE),IN=2,NFREQ)                   WV3D3000
   62 CONTINUE                                                          WV3D3010
```

```
 69 CONTINUE                                                                    WV3D3020
    UAVE=0.0                                                                    WV3D3030
    INDEX=0                                                                     WV3D3040
    IF(IB.EQ.1)RANGE=RANGE+FLOAT(N1)*DX                                         WV3D3050
    IF(RANGE.GT.RANGEM)STOP                                                     WV3D3060
    IF(IT.EQ.1)DEPTH=DEPTH+FLOAT(N3)*DZ                                         WV3D3070
    IF(DEPTH.GT.DEPTHM)STOP                                                     WV3D3080
    STABLE=10.0                                                                 WV3D3090
    II=0                                                                        WV3D3100
    ERROR=0.01                                                                  WV3D3110
    T=0.0                                                                       WV3D3120
    TS12=TSS                                                                    WV3D3130
    IT=1                                                                        WV3D3140
    IS=0                                                                        WV3D3150
    PAUSE 1                                                                     WV3D3160
    GO TO 9                                                                     WV3D3170
 10 FORMAT(8I10)                                                                WV3D3180
 20 FORMAT(0P10E8.1)                                                            WV3D3190
 30 FORMAT(8A10)                                                                WV3D3200
 40 FORMAT(/4H NX=,I2,4H NY=,I2,4H NZ=,I2,4H NT=,I5,5H NXS=,I2,5H NYS=          WV3D3210
   *,I2,5H NZS=,I2,7H NFREQ=,I1)                                                WV3D3220
 50 FORMAT(8H ENERGY=,E14.7,14H AT TIME STEP ,I3,7H, TIME=,E14.7,               WV3D3230
   *6H SEC.,,13H AVE. ENERGY=,E14.7,9H, STABLE=,E14.7)                          WV3D3240
 60 FORMAT(/4H N1=,I2,4H N2=,I2,4H N3=,I2,                    5H N12=,          WV3D3250
   *I5,5H N13=,I5,5H N14=,I5,5H N15=,I5)                                        WV3D3260
 70 FORMAT(/10H VMDV(I,K))                                                      WV3D3270
 80 FORMAT(/10H BMDI(I,K))                                                      WV3D3280
 90 FORMAT(/24H STORAGE SIZE OF ARRAYS=,I10)                                    WV3D3290
100 FORMAT(/(7(1X,I2,1H*,I1,1H*,I2,1X,0P1E10.3)))                              WV3D3300
110 FORMAT( 49H PERCENT ENERGY CHANGE RELATIVE TO LAST ANALYSIS=    ,           WV3D3310
   *0P1E15.8)                                                                   WV3D3320
130 FORMAT(9H VMD(I,K))                                                         WV3D3330
140 FORMAT(9H BMD(I,K))                                                         WV3D3340
150 FORMAT(0P8E10.3)                                                            WV3D3350
160 FORMAT(/4H DX=,0P1E10.3,4H DY=,0P1E10.3,4H DZ=,0P1E10.3)                    WV3D3360
170 FORMAT(1X,0P13E10.3)                                                        WV3D3370
180 FORMAT(/,19H FUNDAMENTAL FREQ.=,E10.3,29H HZ., HARMONIC AMPLITUDES          WV3D3380
   * ARE/(8E10.3))                                                             WV3D3390
200 FORMAT(3(2X,I1,2X,1H,,I2,1H*,I2,1H*,I2,1H*,I1,1X,E10.3,1H,,E10.3))         WV3D3400
210 FORMAT(/8H HX(I,K))                                                         WV3D3410
220 FORMAT(/8H HY(I,K))                                                         WV3D3420
230 FORMAT(/8H HZ(I,K))                                                         WV3D3430
240 FORMAT(/8H EX(I,K))                                                         WV3D3440
250 FORMAT(/8H EY(I,K))                                                         WV3D3450
260 FORMAT(/8H EZ(I,K))                                                         WV3D3460
270 FORMAT(/10H ZFRT(J,K))                                                      WV3D3470
280 FORMAT(/11H ZBACK(J,K))                                                     WV3D3480
290 FORMAT(1H1)                                                                 WV3D3490
```

```
  300 FORMAT(/11H YBACK(J,K))                                                    WV3D3500
  310 FORMAT(//10H ZBOT(I,J))                                                    WV3D3510
  320 FORMAT(/10H ZTOP(I,J))                                                     WV3D3520
  330 FORMAT(/10H YTOP(I,J))                                                     WV3D3530
  340 FORMAT(/11H ZSIDE(I,K))                                                    WV3D3540
  350 FORMAT(/11H YSIDE(I,K))                                                    WV3D3550
  360 FORMAT(/4H DT=,(0P10E10.3)       )                                         WV3D3560
  370 FORMAT(/63H EQUIVALENT AMPLITUDE AND PHASE OF TOP MESHPOINTS FOR BWV3D3570
     *LOCK I7=,I5,5H  ,IX=,I5/                                                   WV3D3580
     *114H HARM  I  J  K L AMP(IJKL) PHASE(IJKL) HARM  I  J  K L AMP(IJKWV3D3590
     *L) PHASE(IJKL) HARM  I  J  K L AMP(IJKL) PHASE(IJKL)          /  )WV3D3600
  380 FORMAT(/64H EQUIVALENT AMPLITUDE AND PHASE OF BACK MESHPOINTS FOR WV3D3610
     *BLOCK I7=,I5,5H  ,IX=,I5/                                                  WV3D3620
     *114H HARM  I  J  K L AMP(IJKL) PHASE(IJKL) HARM  I  J  K L AMP(IJKWV3D3630
     *L) PHASE(IJKL) HARM  I  J  K L AMP(IJKL) PHASE(IJKL)            /)WV3D3640
  390 FORMAT(/23H AVERAGE STORED ENERGY=  ,E15.8)                                WV3D3650
  400 FORMAT(/38H PRESSURE AT LOCATION I*J*K AT TIME T=,E15.8,5H SEC.)           WV3D3660
      END                                                                        WV3D3670


      SUBROUTINE REQ(V,DV)                                                       REQ 0000
C REQ IS THE FIRST ORDER EQUATIONS EQUIVALENT TO THE WAVE EQUATION IN CONTINUOUS
C DISCRETE FORM.
      DIMENSION V(1),DV(1)                                                       REQ 0010
      INTEGER OJK1,OJK2,OPJK1,OPJK12,OJK12,OJK22                                 REQ 0020
      COMMON/SIZE/N1M1,N1,N1P1,N2M1,N2,N2P1,N3M1,N3,N3P1,N12,N13,N14,N15REQ 0030
     *,N23,I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,I11,I12,I13,I14,I15,I16,I17,LTREQ 0040
     *,N132,N133,N142,N143                                                       REQ 0050
      COMMON/TDT/T,DT(20),TS,DX,DY,DZ                                            REQ 0060
      COMMON/MEDIUM/VMD(400),VMDV(400),BMD(400),BMDI(400),FX(400),               REQ 0070
     *FY(400),FZ(400),HX(400),HY(400),HZ(400),ZSIDE(400),YSIDE(400),             REQ 0080
     *ZFRT(100),ZBACK(100),YBACK(100),ZTOP(100),YTOP(100),ZBOT(100)              REQ 0090
C                                                                                REQ 0100
CC V(IJKLM)=V(I+(J-1)*N1+(K-1)*N12+(L-1)*N13+(M-1)*N14)                          REQ 0110
C                                                                                REQ 0120
      DO 10000                                                  K=2,N3M1         REQ 0130
      KN12=K*N12                                                                 REQ 0140
      IJK1=I1+KN12                                                               REQ 0150
      KJ1=IJK1                                                                   REQ 0160
      IK=-N1+K*N1                                                                REQ 0170
      IK1=IK                                                                     REQ 0180
      OJK1=I3+KN12                                                               REQ 0190
      KJ2=OJK1                                                                   REQ 0200
      NJK1=-N12+KN12                                                             REQ 0210
      KJ3=NJK1                                                                   REQ 0220
      NMJK2=I5+KN12                                                              REQ 0230
```

```
      KJ4=NMJK2                                                                  REQ 0240
      JK=-N2+K*N2                                                                REQ 0250
      KJ5=JK                                                                     REQ 0260
      DO 10000                                                  J=2 ,N2M1        REQ 0270
      JN1=J*N1                                                                   REQ 0280
      IJK1=KJ1+JN1                                                               REQ 0290
      JI1=IJK1                                                                   REQ 0300
      OJK1=KJ2+JN1                                                               REQ 0310
      OJK2=OJK1+N13                                                              REQ 0320
      OPJK1=OJK1+1                                                               REQ 0330
      NJK1=KJ3+JN1                                                               REQ 0340
      NMJK2=KJ4+JN1                                                              REQ 0350
      JK=KJ5+J                                                                   REQ 0360
      DV(OJK2)=YBACK(JK)*(V(OJK1)-V(OPJK1))                                      REQ 0370
      DV(OJK1)=-ZBACK(JK)*DV(OJK2)                                               REQ 0380
      DV(NJK1)=ZFRT(JK)*DV(NMJK2)                                                REQ 0390
      DO 10000                                                  I=2,N1M1         REQ 0400
      IJK1=JI1+I                                                                 REQ 0410
      IJK2=IJK1+N13                                                              REQ 0420
      IJK3=IJK2+N13                                                              REQ 0430
      IJK4=IJK3+N13                                                              REQ 0440
      IMJK2=IJK2-1                                                               REQ 0450
      IPJK1=IJK1+1                                                               REQ 0460
      IJMK3=IJK3-N1                                                              REQ 0470
      IJPK1=IJK1+N1                                                              REQ 0480
      IJKP1=IJK1+N12                                                             REQ 0490
      IJKM4=IJK4-N12                                                             REQ 0500
      IK=IK1+I                                                                   REQ 0510
      DV(IJK1)=FX(IK)*(V(IMJK2)-V(IJK2))+FY(IK)*(V(IJMK3)-V(IJK3))+              REQ 0520
     *FZ(IK)*(V(IJKM4)-V(IJK4))                                                  REQ 0530
      DV(IJK2)=HX(IK)*(V(IJK1)-V(IPJK1))                                         REQ 0540
      DV(IJK3)=HY(IK)*(V(IJK1)-V(IJPK1))                                         REQ 0550
      DV(IJK4)=HZ(IK)*(V(IJK1)-V(IJKP1))                                         REQ 0560
10000 CONTINUE                                                                   REQ 0570
      DO 10001                                                  J=2,N2M1         REQ 0580
      IJ11=-N1+J*N1                                                              REQ 0590
      JI1=IJ11                                                                   REQ 0600
      DO 10001                                                  I=2,N1M1         REQ 0610
      IJ11=JI1+I                                                                 REQ 0620
      IJ14=IJ11+N133                                                             REQ 0630
      IJ21=IJ11+N12                                                              REQ 0640
      IJN1=I12+IJ11                                                              REQ 0650
      IJNM4=I13+IJN1                                                             REQ 0660
      DV(IJ14)=YTOP(IJ11)*(V(IJ11)-V(IJ21))                                      REQ 0670
      DV(IJ11)=-ZTOP(IJ11)*DV(IJ14)                                              REQ 0680
      DV(IJN1)=ZBOT(IJ11)*DV(IJNM4)                                              REQ 0690
10001 CONTINUE                                                                   REQ 0700
      DO 10002                                                  K=2,N3M1         REQ 0710
```

```
      I1K1=-N12+K*N12                                                    REQ 0720
      KI1=I1K1                                                           REQ 0730
      IK=-N1+K*N1                                                        REQ 0740
      KI3=IK                                                             REQ 0750
      DO 10002                                          I=2,N1M1         REQ 0760
      I1K1=KI1+I                                                         REQ 0770
      INK1=I11+I1K1                                                      REQ 0780
      INMK3=I10+INK1                                                     REQ 0790
      I2K1=I1K1+N1                                                       REQ 0800
      I1K3=I1K1+N132                                                     REQ 0810
      IK=KI3+I                                                           REQ 0820
      DV(I1K3)=YSIDE(IK)*(V(I1K1 )-V(I2K1 ))                             REQ 0830
      DV(I1K1)=-ZSIDE(IK)*DV(I1K3 )                                      REQ 0840
      DV(INK1)=ZSIDE(IK)*DV(INMK3 )                                      REQ 0850
10002 CONTINUE                                                           REQ 0860
      RETURN                                                             REQ 0870
      ENTRY REQC                                                         REQ 0880
      DO 20000                                          K=2,N3M1         REQ 0890
      KN12=K*N12                                                         REQ 0900
      IJK1=I1+KN12                                                       REQ 0910
      KJ1=IJK1                                                           REQ 0920
      IK=-N1+K*N1                                                        REQ 0930
      IK1=IK                                                             REQ 0940
      OJK1=I3+KN12                                                       REQ 0950
      KJ2=OJK1                                                           REQ 0960
      NJK1=-N12+KN12                                                     REQ 0970
      KJ3=NJK1                                                           REQ 0980
      NMJK2=I5+KN12                                                      REQ 0990
      KJ4=NMJK2                                                          REQ 1000
      JK=-N2+K*N2                                                        REQ 1010
      KJ5=JK                                                             REQ 1020
      DO 20000                                          J=2,N2M1         REQ 1030
      JN1=J*N1                                                           REQ 1040
      IJK1=KJ1+JN1                                                       REQ 1050
      JI1=IJK1                                                           REQ 1060
      OJK1=KJ2+JN1                                                       REQ 1070
      OJK12=OJK1+N14                                                     REQ 1080
      OJK2=OJK1+N13                                                      REQ 1090
      OPJK12=OJK1+1+N14                                                  REQ 1100
      NJK1=KJ3+JN1                                                       REQ 1110
      NMJK2=KJ4+JN1                                                      REQ 1120
      JK=KJ5+J                                                           REQ 1130
      DV(OJK2 )=YBACK(JK)*(V(OJK12)-V(OPJK12))                           REQ 1140
      DV(OJK1 )=-ZBACK(JK)*DV(OJK2)                                      REQ 1150
      DV(NJK1 )=ZFRT(JK)*DV(NMJK2)                                       REQ 1160
      DO 20000                                          I=2,N1M1         REQ 1170
      IJK1=JI1+I                                                         REQ 1180
      IJK12=IJK1+N14                                                     REQ 1190
```

```
      IJK2=IJK1+N13                                                          REQ 1200
      IJK22=IJK2+N14                                                         REQ 1210
      IJK3=IJK2+N13                                                          REQ 1220
      IJK32=IJK3+N14                                                         REQ 1230
      IJK4=IJK3+N13                                                          REQ 1240
      IJK42=IJK4+N14                                                         REQ 1250
      IMJK22=IJK22-1                                                         REQ 1260
      IPJK12=IJK12+1                                                         REQ 1270
      IJMK32=IJK32-N1                                                        REQ 1280
      IJPK12=IJK12+N1                                                        REQ 1290
      IJKP12=IJK12+N12                                                       REQ 1300
      IJKM42=IJK42-N12                                                       REQ 1310
      IK=IK1+I                                                               REQ 1320
      DV(IJK1)=FX(IK)*(V(IMJK22)-V(IJK22))+FY(IK)*(V(IJMK32)-V(IJK32))+      REQ 1330
     *FZ(IK)*(V(IJKM42)-V(IJK42))                                            REQ 1340
      DV(IJK2)=HX(IK)*(V(IJK12)-  V(IPJK12))                                 REQ 1350
      DV(IJK3)=HY(IK)*(V(IJK12)-V(IJPK12))                                   REQ 1360
      DV(IJK4)=HZ(IK)*(V(IJK12)-V(IJKP12))                                   REQ 1370
20000 CONTINUE                                                               REQ 1380
      DO 20001                                          J=2,N2M1             REQ 1390
      JN1=J*N1                                                               REQ 1400
      IJ11=-N1+JN1                                                           REQ 1410
      JI1=IJ11                                                               REQ 1420
      DO 20001                                          I=2,N1M1             REQ 1430
      IJ11=JI1+I                                                             REQ 1440
      IJ112=IJ11+N14                                                         REQ 1450
      IJ14=IJ11+N133                                                         REQ 1460
      IJ212=IJ112+N12                                                        REQ 1470
      IJN1=I12+IJ11                                                          REQ 1480
      IJNM4=I13+IJN1                                                         REQ 1490
      DV(IJ14)=YTOP(IJ11)*(V(IJ112)-V(IJ212))                                REQ 1500
      DV(IJ11)=-ZTOP(IJ11)*DV(IJ14)                                          REQ 1510
      DV(IJN1)=ZBOT(IJ11)*DV(IJNM4)                                          REQ 1520
20001 CONTINUE                                                               REQ 1530
      DO 20002                                          K=2,N3M1             REQ 1540
      I1K1=-N12+K*N12                                                        REQ 1550
      KI1=I1K1                                                               REQ 1560
      IK=-N1+K*N1                                                            REQ 1570
      KI3=IK                                                                 REQ 1580
      DO 20002                                          I=2,N1M1             REQ 1590
      I1K1=KI1+I                                                             REQ 1600
      I1K12=I1K1+N14                                                         REQ 1610
      I1K3=I1K1+N132                                                         REQ 1620
      I2K12=I1K12+N1                                                         REQ 1630
      INK1=I11+I1K1                                                          REQ 1640
      INMK3=I1C+INK1                                                         REQ 1650
      IK=KI3+I                                                               REQ 1660
      DV(I1K3)=YSIDE(IK)*(V(I1K12)-V(I2K12))                                 REQ 1670
```

```
      DV(I1K1)=-ZSIDE(IK)*DV(I1K3)                                          REQ 1680
      DV(INK1)=ZSIDE(IK)*DV(INK3)                                           REQ 1690
20002 CONTINUE                                                              REQ 1700
      RETURN                                                                REQ 1710
      END                                                                   REQ 1720



      SUBROUTINE RKUT(V,DV)                                                 RKUT0000
      DIMENSION V(1),DV(1),CRK(4)                                           RKUT0010
      COMMON/MEDIUM/VMD(400),VMDV(400),BMD(400),BMDI(400),FX(400),          RKUT0020
     *FY(400),FZ(400),HX(400),HY(400),HZ(400),ZSIDE(400),YSIDE(400),        RKUT0030
     *ZFRT(100),ZBACK(100),YBACK(100),ZTOP(100),YTOP(100),ZBOT(100)         RKUT0040
      COMMON/SIZE/N1M1,N1,N1P1,N2M1,N2,N2P1,N3M1,N3,N3P1,N12,N13,N14,N15RKUT0050
     *,N23,I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,I11,I12,I13,I14,I15,I16,I17,LTRKUT0060
     *,N132,N133,N142,N143                                                  RKUT0070
      COMMON/TDT/T,DT(20),TS,DX,DY,DZ                                       RKUT0080
      DATA CRK(1)/0.0/,CRK(2)/0.5/,CRK(3)/0.5/,CRK(4)/1.0/,F16TH/0.16666RKUT0090
     *6666666667/                                                           RKUT0100
C                                                                           RKUT0110
C V(IJKLM)=V(I+(J-1)*N1+(K-1)*N12+(L-1)*N13+(M-1)*N14)                      RKUT0120
C                                                                           RKUT0130
      CALL REQ(V,DV)                                                        RKUT0140
      DO 10001                                          L=1,4               RKUT0150
      IJKL1=I2 +L*N13                                                       RKUT0160
      LK1=IJKL1                                                             RKUT0170
      DO 10001                                          K=1,N3              RKUT0180
      IJKL1=LK1+K*N12                                                       RKUT0190
      KJ1=IJKL1                                                             RKUT0200
      DO 10001                                          J=1,N2              RKUT0210
      IJKL1=KJ1+J*N1                                                        RKUT0220
      JI1=IJKL1                                                             RKUT0230
      DO 10001                                          I=1,N1              RKUT0240
      IJKL1=JI1+I                                                           RKUT0250
      IJKL3=IJKL1+N142                                                      RKUT0260
      V(IJKL3)=DT(LT)*DV(IJKL1)                                             RKUT0270
10001 CONTINUE                                                              RKUT0280
      DO 10003                                          M=4,6               RKUT0290
      CRKD=CRK(M-2)                                                         RKUT0300
      MN14=M*N14                                                            RKUT0310
      MMN14=-N142+MN14                                                      RKUT0320
      IJKLM=-N14+MN14                                                       RKUT0330
      ML1=IJKLM                                                             RKUT0340
      DO 10002                                          L=1,4               RKUT0350
      IJKL1=I2 +L*N13                                                       RKUT0360
      LK1=IJKL1                                                             RKUT0370
      DO 10002                                          K=1,N3              RKUT0380
```

```
      IJKL1=LK1+K*N12                                                    RKUT0390
      KJ1=IJKL1                                                          RKUT0400
      DO 10002                                          J=1,N2           RKUT0410
      IJKL1=KJ1+J*N1                                                     RKUT0420
      JI1=IJKL1                                                          RKUT0430
      DO 10002                                          I=1,N1           RKUT0440
      IJKL1=JI1+I                                                        RKUT0450
      IJKL2=IJKL1+N14                                                    RKUT0460
      IJKLMM=IJKL1+MMN14                                                 RKUT0470
      V(IJKL2)=V(IJKL1)+CRKD*V(IJKLMM)                                   RKUT0480
10002 CONTINUE                                                           RKUT0490
      CALL REQC(V,DV)                                                    RKUT0500
      DO 10003                                          L=1,4            RKUT0510
      IJKL1=I2 +L*N13                                                    RKUT0520
      LK1=IJKL1                                                          RKUT0530
      DO 10003                                          K=1,N3           RKUT0540
      IJKL1=LK1+K*N12                                                    RKUT0550
      KJ1=IJKL1                                                          RKUT0560
      DO 10003                                          J=1,N2           RKUT0570
      IJKL1=KJ1+J*N1                                                     RKUT0580
      JI1=IJKL1                                                          RKUT0590
      DO 10003                                          I=1,N1           RKUT0600
      IJKL1=JI1+I                                                        RKUT0610
      IJKLM=ML1+IJKL1                                                    RKUT0620
      V(IJKLM)=DT(LT)*DV(IJKL1)                                          RKUT0630
10003 CONTINUE                                                           RKUT0640
      DO 10004                                          L=1,4            RKUT0650
      IJKL1=I2 +L*N13                                                    RKUT0660
      LK1=IJKL1                                                          RKUT0670
      DO 10004                                          K=1,N3           RKUT0680
      IJKL1=LK1+K*N12                                                    RKUT0690
      KJ1=IJKL1                                                          RKUT0700
      DO 10004                                          J=1,N2           RKUT0710
      IJKL1=KJ1+J*N1                                                     RKUT0720
      JI1=IJKL1                                                          RKUT0730
      DO 10004                                          I=1,N1           RKUT0740
      IJKL1=JI1+I                                                        RKUT0750
      IJKL3=IJKL1+N142                                                   RKUT0760
      IJKL4=IJKL3+N14                                                    RKUT0770
      IJKL5=IJKL4+N14                                                    RKUT0780
      IJKL6=IJKL5+N14                                                    RKUT0790
      V(IJKL1)=V(IJKL1)+(V(IJKL3)+2.0*(V(IJKL4)+V(IJKL5))+V(IJKL6))*     RKUT0800
     *F16TH                                                              RKUT0810
10004 CONTINUE                                                           RKUT0820
      RETURN                                                             RKUT0830
      END                                                                RKUT0840
```

```
      FUNCTION U(V)                                                          U    0000
      INTEGER CK,CJK2                                                        U    0010
      DIMENSION V(1)                                                         U    0020
      COMMON/MEDIUM/VMD(400),VMDV(400),BMD(400),BMDI(400),FX(400),           U    0030
     *FY(400),FZ(400),HX(400),HY(400),HZ(400),ZSIDE(400),YSIDE(400),         U    0040
     *ZFRT(100),ZBACK(100),YBACK(100),ZTOP(100),YTOP(100),ZBOT(100)          U    0050
      COMMON/SIZE/N1M1,N1,N1P1,N2M1,N2,N2P1,N3M1,N3,N3P1,N12,N13,N14,N15U    0060
     *,N23,I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,I11,I12,I13,I14,I15,I16,I17,LTU    0070
     *,N132,N133,N142,N143                                                  U    0080
      COMMON/TDT/T,DT(20),TS,DX,DY,DZ                                        U    0090
C                                                                            U    0100
CC V(IJKLM)=V(I+(J-1)*N1+(K-1)*N12+(L-1)*N13+(M-1)*N14)                      U    0110
C                                                                            U    0120
      EN=0.                                                                  U    0130
      DO 1                                                    K=2,N3M1       U    0140
      IJK1=I1+K*N12                                                          U    0150
      KJ5=IJK1                                                               U    0160
      IK=-N1+K*N1                                                            U    0170
      KI1=IK                                                                 U    0180
      DO 1                                                    J=2,N2M1       U    0190
      IJK1=KJ5+J*N1                                                          U    0200
      JI5=IJK1                                                               U    0210
      DO 1                                                    I=2,N1M1       U    0220
      IJK1=JI5+I                                                             U    0230
      IJK2=IJK1+N13                                                          U    0240
      IJK3=IJK2+N13                                                          U    0250
      IJK4=IJK3+N13                                                          U    0260
      IK=KI1+I                                                               U    0270
      EN=EN+VMDV(IK)*(V(IJK2)**2+V(IJK3)**2+V(IJK4)**2)                      U    0280
      EN=EN+BMDI(IK)*V(IJK1)**2                                              U    0290
    1 CONTINUE                                                               U    0300
      DO 2                                                    J=2,N2M1       U    0310
      IJ=-N1+J*N1                                                            U    0320
      JI=IJ                                                                  U    0330
      DO 2                                                    I=2,N1M1       U    0340
      IJ=JI+I                                                                U    0350
      IJ14=IJ+N133                                                           U    0360
      EN=EN+VMDV(I )*V(IJ14)**2                                              U    0370
    2 CONTINUE                                                               U    0380
      DO 3                                                    K=2,N3M1       U    0390
      IK=-N1+K*N1                                                            U    0400
      KI=IK                                                                  U    0410
      I1K3=I7+K*N12                                                          U    0420
      KI1=I1K3                                                               U    0430
      DO 3                                                    I=2,N1M1       U    0440
      IK=KI+I                                                                U    0450
      I1K3=KI1+I                                                             U    0460
      EN=EN+VMDV(IK)*V(I1K3)**2                                              U    0470
```

```
    3 CONTINUE                                                                  U    0480
      DO 4                                                  K=2,N3M1            U    0490
      OK=1-N1+K*N1                                                              U    0500
      OJK2=I4+K*N12                                                             U    0510
      KJ=OJK2                                                                   U    0520
      DO 4                                                  J=2,N2M1            U    0530
      OJK2=KJ+J*N1                                                              U    0540
      EN=EN+VMDV(OK)*V(OJK2)**2                                                 U    0550
    4 CONTINUE                                                                  U    0560
      U=EN                                                                      U    0570
      RETURN                                                                    U    0580
      END                                                                       U    0590



      SUBROUTINE TOP(V,C,FUN)                                                   TOP  0000
      DIMENSION V(1),C( 8,600)                                                  TOP  0010
      COMMON/SIZE/N1M1,N1,N1P1,N2M1,N2,N2P1,N3M1,N3,N3P1,N12,N13,N14,N15TOP  0020
     *,N23,I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,I11,I12,I13,I14,I15,I16,I17,LTTOP  0030
     *,N132,N133,N142,N143                                                      TOP  0040
      COMMON/SIG/NXYZS,NFREQ                                                    TOP  0050
      COMMON/TOT/T,DT(20),TS,DX,DY,DZ                                           TOP  0060
C                                                                               TOP  0070
C  V(IJKLM)=V(I+(J-1)*N1+(K-1)*N12+(L-1)*N13+(M-1)*N14)                         TOP  0080
C                                                                               TOP  0090
      PI2T=6.2831853*T                                                          TOP  0100
      DO 1 L=1,4                                                                TOP  0110
      IJ2L=I8+L*N13                                                             TOP  0120
      LJ=IJ2L                                                                   TOP  0130
      IJLE=I1+L*N12                                                             TOP  0140
      LJ1=IJLE                                                                  TOP  0150
      DO 1 J=2,N2M1                                                             TOP  0160
      JN1=J*N1                                                                  TOP  0170
      IJ2L=LJ+JN1                                                               TOP  0180
      IJLE=LJ1+JN1                                                              TOP  0190
      JI=IJ2L                                                                   TOP  0200
      JI1=IJLE                                                                  TOP  0210
      DO 1 I=2,N1M1                                                             TOP  0220
      IJ2L=JI+I                                                                 TOP  0230
      IJLE=2*(JI1+I)                                                            TOP  0240
      IJLO=IJLE-1                                                               TOP  0250
      V(IJ2L)=0.0                                                               TOP  0260
      DO 1 IN=2,NFREQ                                                           TOP  0270
      FIN=IN-1                                                                  TOP  0280
      V(IJ2L)=V(IJ2L)+ C(IN,IJLO)*COS(PI2T*FIN*FUN  +C(IN,IJLE))                TOP  0290
    1 CONTINUE                                                                  TOP  0300
      RETURN                                                                    TOP  0310
```

```
      END                                                                TOP 0320


      SUBROUTINE SOURCE(V,AMP,FUN)                                       SOUR0000
      DIMENSION V(1),AMP(8)                                              SOUR0010
C   THIS SUBROUTINE PRODUCES THE SIGNAL VS. TIME AT LOCATION NXS,NYS,NZS.SOUR0020
C   THE FREQUENCIES OF THE SIGNAL ARE FUN,2*FUN,3*FUN,..,NFREQ         SOUR0030
      COMMON/TDT/T,DT(20),TS,DX,DY,DZ                                    SOUR0040
      COMMON/SIG/NXYZS,NFREQ                                             SOUR0050
      COMMON/SIZE/N1M1,N1,N1P1,N2M1,N2,N2P1,N3M1,N3,N3P1,N12,N13,N14,N15SOUR0060
     *,N23,I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,I11,I12,I13,I14,I15,I16,I17,LTSOUR0070
     *,N132,N133,N142,N143                                              SOUR0080
      PI2T=6.2831853*T                                                   SOUR0090
      V(NXYZS)=0.0                                                       SOUR0100
      DO 1 I=1,NFREQ                                                     SOUR0110
      FI=I                                                               SOUR0120
    1 V(NXYZS)=V(NXYZS)+AMP(I)*COS(PI2T*FI*FUN)                          SOUR0130
      RETURN                                                             SOUR0140
      END                                                                SOUR0150


      SUBROUTINE BACK (V,B,FUN)                                          BACK0000
      DIMENSION V(1),B( 8,600)                                           BACK0010
      INTEGER OPJKL                                                      BACK0020
      COMMON/SIZE/N1M1,N1,N1P1,N2M1,N2,N2P1,N3M1,N3,N3P1,N12,N13,N14,N15BACK0030
     *,N23,I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,I11,I12,I13,I14,I15,I16,I17,LTBACK0040
     *,N132,N133,N142,N143                                              BACK0050
      COMMON/TDT/T,DT(20),TS,DX,DY,DZ                                    BACK0060
      COMMON/SIG/NXYZS,NFREQ                                             BACK0070
C                                                                        BACK0080
CC V(IJKLM)=V(I+(J-1)*N1+(K-1)*N12+(L-1)*N13+(M-1)*N14)                  BACK0090
C                                                                        BACK0100
      PI2T=6.2831853*T                                                   BACK0110
      DO 1 L=1,4                                                         BACK0120
      OPJKL=I9+L*N13                                                     BACK0130
      LK =OPJKL                                                          BACK0140
      JKLE=I6+L*N23                                                      BACK0150
      LK1=JKLE                                                           BACK0160
      DO 1 K=2,N3M1                                                      BACK0170
      OPJKL=LK+K*N12                                                     BACK0180
      KJ=OPJKL                                                           BACK0190
      JKLE=LK1+K*N2                                                      BACK0200
      KJ1=JKLE                                                           BACK0210
      DO 1 J=2,N2M1                                                      BACK0220
```

```
      OPJKL=KJ+J*N1                                                      BACK0230
      JKLE=2*(KJ1+J)                                                     BACK0240
      JKLO=JKLE-1                                                        BACK0250
      V(OPJKL)=0.0                                                       BACK0260
      DO 1 IN=2,NFREQ                                                    BACK0270
      FIN=IN-1                                                           BACK0280
      V(OPJKL)=V(OPJKL)+B(IN,JKLO)*COS(PI2T*FIN*FUN +B(IN,JKLE))         BACK0290
    1 CONTINUE                                                           BACK0300
      RETURN                                                             BACK0310
      END                                                                BACK0320
C     ..................................................................HARM0000
C                                                                        HARM0010
C        SUBROUTINE HARM                                                 HARM0020
C                                                                        HARM0030
C        PURPOSE                                                         HARM0040
C           PERFORMS DISCRETE COMPLEX FOURIER TRANSFORMS ON A COMPLEX    HARM0050
C           THREE DIMENSIONAL ARRAY                                      HARM0060
C                                                                        HARM0070
C           CALL HARM (A,M,INV,S,IFSET,IFERR)                            HARM0080
C        USAGE                                                           HARM0090
C                                                                        HARM0100
C        DESCRIPTION OF PARAMETERS                                       HARM0110
C           A      - AS INPUT, A CONTAINS THE COMPLEX, 3-DIMENSIONAL     HARM0120
C                    ARRAY TO BE TRANSFORMED.  THE REAL PART OF          HARM0130
C                    A(I1,I2,I3) IS STORED IN VECTOR FASHION IN A CELL   HARM0140
C                    WITH INDEX 2*(I3*N1*N2 + I2*N1 + I1) + 1 WHERE      HARM0150
C                    NI = 2**M(I), I=1,2,3 AND I1 = 0,1,...,N1-1 ETC.    HARM0160
C                    THE IMAGINARY PART IS IN THE CELL IMMEDIATELY       HARM0170
C                    FOLLOWING.  NOTE THAT THE SUBSCRIPT I1 INCREASES    HARM0180
C                    MOST RAPIDLY AND I3 INCREASES LEAST RAPIDLY.        HARM0190
C                    AS OUTPUT, A CONTAINS THE COMPLEX FOURIER           HARM0200
C                    TRANSFORM.  THE NUMBER OF CORE LOCATIONS OF         HARM0210
C                    ARRAY A IS 2*(N1*N2*N3)                             HARM0220
C           M      - A THREE CELL VECTOR WHICH DETERMINES THE SIZES      HARM0230
C                    OF THE 3 DIMENSIONS OF THE ARRAY A.   THE SIZE,     HARM0240
C                    NI, OF THE I DIMENSION OF A IS 2**M(I), I = 1,2,3   HARM0250
C           INV    - A VECTOR WORK AREA FOR BIT AND INDEX MANIPULATION   HARM0260
C                    OF DIMENSION ONE EIGHTH THE NUMBER OF CORE          HARM0270
C                    LOCATIONS OF A, VIZ., (1/8)*2*N1*N2*N3              HARM0280
C           S      - A VECTOR WORK AREA FOR SINE TABLES WITH DIMENSION   HARM0290
C                    THE SAME AS INV                                     HARM0300
C           IFSET  - AN OPTION PARAMETER WITH THE FOLLOWING SETTINGS     HARM0310
C                       0     SET UP SINE AND INV TABLES ONLY           HARM0320
C                       1     SET UP SINE AND INV TABLES ONLY AND       HARM0330
C                             CALCULATE FOURIER TRANSFORM               HARM0340
C                      -1     SET UP SINE AND INV TABLES ONLY AND       HARM0350
C                             CALCULATE INVERSE FOURIER TRANSFORM (FOR  HARM0360
C                             THE MEANING OF INVERSE SEE THE EQUATIONS  HARM0370
```

```
C                        UNDER METHOD BELOW)                                    HARM0380
C                    2   CALCULATE FOURIER TRANSFORM ONLY (ASSUME              HARM0390
C                        SINE AND INV TABLES EXIST)                             HARM0400
C                   -2   CALCULATE INVERSE FOURIER TRANSFORM ONLY               HARM0410
C                        (ASSUME SINE AND INV TABLES EXIST)                     HARM0420
C        IFERR - ERROR INDICATOR.   WHEN IFSET IS 0,+1,-1,                      HARM0430
C                IFERR = 1 MEANS THE MAXIMUM M(I) IS LESS THAN 3                HARM0440
C                OR GREATER THAN 20, I=1,2,3  WHEN IFSET IS                     HARM0450
C                +2,-2, IFERR = 1 MEANS THAT THE SINE AND INV                   HARM0460
C                TABLES ARE NOT LARGE ENOUGH OR HAVE NOT BEEN                   HARM0470
C                COMPUTED.  IF ON RETURN IFERR =0 THEN NONE OF                  HARM0480
C                THE ABOVE CONDITIONS ARE PRESENT                               HARM0490
C                                                                               HARM0500
C     REMARKS                                                                   HARM0510
C        THIS SUBROUTINE IS TO BE USED FOR COMPLEX, 3-DIMENSIONAL               HARM0520
C        ARRAYS IN WHICH EACH DIMENSION IS A POWER OF 2.  THE                   HARM0530
C        MAXIMUM M(I) MUST NOT BE LESS THAN 3 OR GREATER THAN 20,               HARM0540
C        I = 1,2,3                                                              HARM0550
C                                                                               HARM0560
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED                             HARM0570
C        NONE                                                                   HARM0580
C                                                                               HARM0590
C     METHOD                                                                    HARM0600
C        FOR IFSET = +1, OR +2, THE FOURIER TRANSFORM OF COMPLEX                HARM0610
C        ARRAY A IS OBTAINED.                                                   HARM0620
C                                                                               HARM0630
C                    N1-1   N2-1   N3-1                    L1   L2   L3         HARM0640
C     X(J1,J2,J3)=SUM     SUM    SUM     A(K1,K2,K3)*W1  *W2  *W3              HARM0650
C                 K1=0   K2=0   K3=0                                            HARM0660
C                                                                               HARM0670
C                 WHERE WI IS THE N(I) ROOT OF UNITY AND L1=K1*J1,              HARM0680
C                       L2=K2*J2, L3=K3*J3                                      HARM0690
C                                                                               HARM0700
C                                                                               HARM0710
C        FOR IFSET = -1, OR -2, THE INVERSE FOURIER TRANSFORM A OF              HARM0720
C        COMPLEX ARRAY X IS OBTAINED.                                           HARM0730
C                                                                               HARM0740
C     A(K1,K2,K3)=                                                              HARM0750
C              1       N1-1   N2-1   N3-1                   -L1  -L2  -L3       HARM0760
C        -------- *SUM     SUM    SUM     X(K1,K2,K3)*W1  *W2  *W3            HARM0770
C        N1*N2*N3   J1=0   J2=0   J3=0                                          HARM0780
C                                                                               HARM0790
C                                                                               HARM0800
C        SEE J.W. COOLEY AND J.W. TUKEY, "AN AN ALGORITHM FOR THE               HARM0810
C        MACHINE CALCULATION OF COMPLEX FOURIER SERIES",                        HARM0820
C        MATHEMATICS OF COMPUTATIONS, VOL. 19 (APR. 1965), P. 297.              HARM0830
C                                                                               HARM0840
C     ..........................................................................HARM0850
```

```
      SUBROUTINE HARM(A,M,INV,S,IFSET, IFERR)                          HARM0860
      DIMENSION A(1),INV(1),S(1),N(3),M(3),NP(3),W(2),W2(2),W3(2)      HARM0870
      EQUIVALENCE (N1,N(1)),(N2,N(2)),(N3,N(3))                        HARM0880
   10 IF( IABS(IFSET) - 1) 900,900,12                                  HARM0890
   12 MTT=MAXO(M(1),M(2),M(3)) -2                                      HARM0900
      ROOT2 = SQRT(2.)                                                 HARM0910
      IF (MTT-MT ) 14,14,13                                            HARM0920
   13 IFERR=1                                                          HARM0930
      RETURN                                                           HARM0940
   14 IFERR=0                                                          HARM0950
      M1=M(1)                                                          HARM0960
      M2=M(2)                                                          HARM0970
      M3=M(3)                                                          HARM0980
      N1=2**M1                                                         HARM0990
      N2=2**M2                                                         HARM1000
      N3=2**M3                                                         HARM1010
   16 IF(IFSET) 18,18,20                                               HARM1020
   18 NX= N1*N2*N3                                                     HARM1030
      FN = NX                                                          HARM1040
      DO 19 I = 1,NX                                                   HARM1050
      A(2*I-1) = A(2*I-1)/FN                                           HARM1060
   19 A(2*I) = -A(2*I)/FN                                              HARM1070
   20 NP(1)=N1*2                                                       HARM1080
      NP(2)= NP(1)*N2                                                  HARM1090
      NP(3)=NP(2)*N3                                                   HARM1100
      DO 250 ID=1,3                                                    HARM1110
      IL = NP(3)-NP(ID)                                                HARM1120
      IL1 = IL+1                                                       HARM1130
      MI = M(ID)                                                       HARM1140
      IF (MI)250,250,30                                                HARM1150
   30 IDIF=NP(ID)                                                      HARM1160
      KBIT=NP(ID)                                                      HARM1170
      MEV = 2*(MI/2)                                                   HARM1180
      IF (MI - MEV )60,60,40                                           HARM1190
C                                                                      HARM1200
C     M IS ODD. DO L=1 CASE                                            HARM1210
   40 KBIT=KBIT/2                                                      HARM1220
      KL=KBIT-2                                                        HARM1230
      DO 50 I=1,IL1,IDIF                                               HARM1240
      KLAST=KL+I                                                       HARM1250
      DO 50 K=I,KLAST,2                                                HARM1260
      KD=K+KBIT                                                        HARM1270
C                                                                      HARM1280
C     DO ONE STEP WITH L=1,J=0                                         HARM1290
```

```
C      A(K)=A(K)+A(KD)                                                  HARM1300
CC     A(KD)=A(K)-A(KD)                                                 HARM1310
C                                                                       HARM1320
       T=A(KD)                                                          HARM1330
       A(KD)=A(K)-T                                                     HARM1340
       A(K)=A(K)+T                                                      HARM1350
       T=A(KD+1)                                                        HARM1360
       A(KD+1)=A(K+1)-T                                                 HARM1370
   50  A(K+1)=A(K+1)+T                                                  HARM1380
       IF (MI - 1)250,250,52                                            HARM1390
   52  LFIRST =3                                                        HARM1400
C                                                                       HARM1410
C      DEF - JLAST = 2**(L-2) -1                                        HARM1420
       JLAST=1                                                          HARM1430
       GO TO 70                                                         HARM1440
C                                                                       HARM1450
C      M IS EVEN                                                        HARM1460
   60  LFIRST = 2                                                       HARM1470
       JLAST=0                                                          HARM1480
   70  DO 240 L=LFIRST,MI,2                                             HARM1490
       JJDIF=KBIT                                                       HARM1500
       KBIT=KBIT/4                                                      HARM1510
       KL=KBIT-2                                                        HARM1520
C                                                                       HARM1530
C      DO FOR J=0                                                       HARM1540
       DO 80 I=1,IL1,IDIF                                               HARM1550
       KLAST=I+KL                                                       HARM1560
       DO 80 K=I,KLAST,2                                                HARM1570
       K1=K+KBIT                                                        HARM1580
       K2=K1+KBIT                                                       HARM1590
       K3=K2+KBIT                                                       HARM1600
C                                                                       HARM1610
C      DO TWO STEPS WITH J=0                                            HARM1620
C      A(K)=A(K)+A(K2)                                                  HARM1630
C      A(K2)=A(K)-A(K2)                                                 HARM1640
C      A(K1)=A(K1)+A(K3)                                                HARM1650
C      A(K3)=A(K1)-A(K3)                                                HARM1660
C                                                                       HARM1670
C      A(K)=A(K)+A(K1)                                                  HARM1680
C      A(K1)=A(K)-A(K1)                                                 HARM1690
C      A(K2)=A(K2)+A(K3)*I                                              HARM1700
C      A(K3)=A(K2)-A(K3)*I                                              HARM1710
C                                                                       HARM1720
       T=A(K2)                                                          HARM1730
       A(K2)=A(K)-T                                                     HARM1740
       A(K)=A(K)+T                                                      HARM1750
       T=A(K2+1)                                                        HARM1760
       A(K2+1)=A(K+1)-T                                                 HARM1770
```

```
      A(K+1)=A(K+1)+T                                                    HARM1780
C                                                                        HARM1790
      T=A(K3)                                                            HARM1800
      A(K3)=A(K1)-T                                                      HARM1810
      A(K1)=A(K1)+T                                                      HARM1820
      T=A(K3+1)                                                          HARM1830
      A(K3+1)=A(K1+1)-T                                                  HARM1840
      A(K1+1)=A(K1+1)+T                                                  HARM1850
C                                                                        HARM1860
      T=A(K1)                                                            HARM1870
      A(K1)=A(K)-T                                                       HARM1880
      A(K)=A(K)+T                                                        HARM1890
      T=A(K1+1)                                                          HARM1900
      A(K1+1)=A(K+1)-T                                                   HARM1910
      A(K+1)=A(K+1)+T                                                    HARM1920
C                                                                        HARM1930
      R=-A(K3+1)                                                         HARM1940
      T = A(K3)                                                          HARM1950
      A(K3)=A(K2)-R                                                      HARM1960
      A(K2)=A(K2)+R                                                      HARM1970
      A(K3+1)=A(K2+1)-T                                                  HARM1980
   80 A(K2+1)=A(K2+1)+T                                                  HARM1990
      IF (JLAST) 235,235,82                                              HARM2000
   82 JJ=JJDIF   +1                                                      HARM2010
C                                                                        HARM2020
CC    DO FOR J=1                                                         HARM2030
      ILAST= IL +JJ                                                      HARM2040
      DO 85 I = JJ,ILAST,IDIF                                            HARM2050
      KLAST = KL+I                                                       HARM2060
      DO 85 K=I,KLAST,2                                                  HARM2070
      K1 = K+KBIT                                                        HARM2080
      K2 = K1+KBIT                                                       HARM2090
      K3 = K2+KBIT                                                       HARM2100
C                                                                        HARM2110
CCC   LETTING W=(1+I)/ROOT2,W3=(-1+I)/ROOT2,W2=I,                        HARM2120
CCC   A(K)=A(K)+A(K2)*I                                                  HARM2130
CCC   A(K2)=A(K)-A(K2)*I                                                 HARM2140
CCC   A(K1)=A(K1)*W+A(K3)*W3                                             HARM2150
CCC   A(K3)=A(K1)*W-A(K3)*W3                                             HARM2160
CCC                                                                      HARM2170
CCC   A(K)=A(K)+A(K1)                                                    HARM2180
CCC   A(K1)=A(K)-A(K1)                                                   HARM2190
CCC   A(K2)=A(K2)+A(K3)*I                                                HARM2200
CCC   A(K3)=A(K2)-A(K3)*I                                                HARM2210
CC                                                                       HARM2220
      R =-A(K2+1)                                                        HARM2230
      T = A(K2)                                                          HARM2240
      A(K2) = A(K)-R                                                     HARM2250
```

```
      A(K) = A(K)+R                                                     HARM2260
      A(K2+1)=A(K+1)-T                                                  HARM2270
      A(K+1)=A(K+1)+T                                                   HARM2280
C                                                                       HARM2290
      AWR=A(K1)-A(K1+1)                                                 HARM2300
      AWI = A(K1+1)+A(K1)                                               HARM2310
      R=-A(K3)-A(K3+1)                                                  HARM2320
      T=A(K3)-A(K3+1)                                                   HARM2330
      A(K3)=(AWR-R)/ROOT2                                               HARM2340
      A(K3+1)=(AWI-T)/ROOT2                                             HARM2350
      A(K1)=(AWR+R)/ROOT2                                               HARM2360
      A(K1+1)=(AWI+T)/ROOT2                                             HARM2370
      T= A(K1)                                                          HARM2380
      A(K1)=A(K)-T                                                      HARM2390
      A(K)=A(K)+T                                                       HARM2400
      T=A(K1+1)                                                         HARM2410
      A(K1+1)=A(K+1)-T                                                  HARM2420
      A(K+1)=A(K+1)+T                                                   HARM2430
      R=-A(K3+1)                                                        HARM2440
      T=A(K3)                                                           HARM2450
      A(K3)=A(K2)-R                                                     HARM2460
      A(K2)=A(K2)+R                                                     HARM2470
      A(K3+1)=A(K2+1)-T                                                 HARM2480
   85 A(K2+1)=A(K2+1)+T                                                 HARM2490
      IF(JLAST-1) 235,235,90                                            HARM2500
   90 JJ= JJ + JJDIF                                                    HARM2510
C                                                                       HARM2520
C     NOW DO THE REMAINING J'S                                          HARM2530
      DO 230 J=2,JLAST                                                  HARM2540
C                                                                       HARM2550
C     FETCH W'S                                                         HARM2560
C     DEF- W=W**INV(J), W2=W**2, W3=W**3                                HARM2570
   96 I=INV(J+1)                                                        HARM2580
   98 IC=NT-I                                                           HARM2590
      W(1)=S(IC)                                                        HARM2600
      W(2)=S(I)                                                         HARM2610
      I2=2*I                                                            HARM2620
      I2C=NT-I2                                                         HARM2630
      IF(I2C)120,110,100                                                HARM2640
C                                                                       HARM2650
C     2*I IS IN FIRST QUADRANT                                          HARM2660
  100 W2(1)=S(I2C)                                                      HARM2670
      W2(2)=S(I2)                                                       HARM2680
      GO TO 130                                                         HARM2690
  110 W2(1)=0.                                                          HARM2700
      W2(2)=1.                                                          HARM2710
      GO TO 130                                                         HARM2720
C                                                                       HARM2730
```

```
C       2*I  IS IN SECOND QUADRANT                                      HARM2740
  120   I2CC  = I2C+NT                                                 HARM2750
        I2C=-I2C                                                        HARM2760
        W2(1)=-S(I2C)                                                   HARM2770
        W2(2)=S(I2CC)                                                   HARM2780
  130   I3=I+I2                                                         HARM2790
        I3C=NT-I3                                                       HARM2800
        IF(I3C)160,150,140                                              HARM2810
C                                                                       HARM2820
C       I3 IN FIRST QUADRANT                                            HARM2830
  140   W3(1)=S(I3C)                                                    HARM2840
        W3(2)=S(I3)                                                     HARM2850
        GO TO 200                                                       HARM2860
  150   W3(1)=0.                                                        HARM2870
        W3(2)=1.                                                        HARM2880
        GO TO 200                                                       HARM2890
C                                                                       HARM2900
  160   I3CC=I3C+NT                                                     HARM2910
        IF(I3CC)190,180,170                                             HARM2920
C                                                                       HARM2930
C       I3 IN SECOND QUADRANT                                           HARM2940
  170   I3C=-I3C                                                        HARM2950
        W3(1)=-S(I3C)                                                   HARM2960
        W3(2)=S(I3CC)                                                   HARM2970
        GO TO 200                                                       HARM2980
  180   W3(1)=-1.                                                       HARM2990
        W3(2)=0.                                                        HARM3000
        GO TO 200                                                       HARM3010
C                                                                       HARM3020
C       3*I  IN THIRD QUADRANT                                          HARM3030
  190   I3CCC=NT+I3CC                                                   HARM3040
        I3CC  = -I3CC                                                   HARM3050
        W3(1)=-S(I3CCC)                                                 HARM3060
        W3(2)=-S(I3CC)                                                  HARM3070
  200   ILAST=IL+JJ                                                     HARM3080
        DO 220 I=JJ,ILAST,IDIF                                          HARM3090
        KLAST=KL+I                                                      HARM3100
        DO 220 K=I,KLAST,2                                              HARM3110
        K1=K+KBIT                                                       HARM3120
        K2=K1+KBIT                                                      HARM3130
        K3=K2+KBIT                                                      HARM3140
C                                                                       HARM3150
CCC     DO TWO STEPS WITH J NOT 0                                       HARM3160
CCC     A(K)=A(K)+A(K2)*W2                                              HARM3170
CCC     A(K2)=A(K)-A(K2)*W2                                             HARM3180
CCC     A(K1)=A(K1)*W+A(K3)*W3                                          HARM3190
CCC     A(K3)=A(K1)*W-A(K3)*W3                                          HARM3200
C                                                                       HARM3210
```

```
C       A(K)=A(K)+A(K1)                                                 HARM3220
C       A(K1)=A(K)-A(K1)                                                HARM3230
C       A(K2)=A(K2)+A(K3)*I                                             HARM3240
C       A(K3)=A(K2)-A(K3)*I                                             HARM3250
C                                                                       HARM3260
        R=A(K2)*W2(1)-A(K2+1)*W2(2)                                     HARM3270
        T=A(K2)*W2(2)+A(K2+1)*W2(1)                                     HARM3280
        A(K2)=A(K)-R                                                    HARM3290
        A(K)=A(K)+R                                                     HARM3300
        A(K2+1)=A(K+1)-T                                                HARM3310
        A(K+1)=A(K+1)+T                                                 HARM3320
C                                                                       HARM3330
        R=A(K3)*W3(1)-A(K3+1)*W3(2)                                     HARM3340
        T=A(K3)*W3(2)+A(K3+1)*W3(1)                                     HARM3350
        AWR=A(K1)*W(1)-A(K1+1)*W(2)                                     HARM3360
        AWI=A(K1)*W(2)+A(K1+1)*W(1)                                     HARM3370
        A(K3)=AWR-R                                                     HARM3380
        A(K3+1)=AWI-T                                                   HARM3390
        A(K1)=AWR+R                                                     HARM3400
        A(K1+1)=AWI+T                                                   HARM3410
        T=A(K1)                                                         HARM3420
        A(K1)=A(K)-T                                                    HARM3430
        A(K)=A(K)+T                                                     HARM3440
        T=A(K1+1)                                                       HARM3450
        A(K1+1)=A(K+1)-T                                                HARM3460
        A(K+1)=A(K+1)+T                                                 HARM3470
        R=-A(K3+1)                                                      HARM3480
        T=A(K3)                                                         HARM3490
        A(K3)=A(K2)-R                                                   HARM3500
        A(K2)=A(K2)+R                                                   HARM3510
        A(K3+1)=A(K2+1)-T                                               HARM3520
  220   A(K2+1)=A(K2+1)+T                                               HARM3530
C       END OF I AND K LOOPS                                            HARM3540
C                                                                       HARM3550
  230   JJ=JJDIF+JJ                                                     HARM3560
C       END OF J-LOOP                                                   HARM3570
C                                                                       HARM3580
  235   JLAST=4*JLAST+3                                                 HARM3590
  240   CONTINUE                                                        HARM3600
C       END OF  L  LOOP                                                 HARM3610
C                                                                       HARM3620
  250   CONTINUE                                                        HARM3630
C       END OF  ID  LOOP                                                HARM3640
C                                                                       HARM3650
C       WE NOW HAVE THE COMPLEX FOURIER SUMS BUT THEIR ADDRESSES ARE    HARM3660
C       BIT-REVERSED.  THE FOLLOWING ROUTINE PUTS THEM IN ORDER         HARM3670
        NTSQ=NT*NT                                                      HARM3680
        M3MT=M3-MT                                                      HARM3690
```

```
  350 IF(M3MT) 370,360,360                                          HARM3700
C                                                                   HARM3710
C     M3 GR. OR EQ. MT                                              HARM3720
  360 IGO3=1                                                        HARM3730
      N3VNT=N3/NT                                                   HARM3740
      MINN3=NT                                                      HARM3750
      GO TO 380                                                     HARM3760
C                                                                   HARM3770
C     M3 LESS THAN MT                                               HARM3780
  370 IGO3=2                                                        HARM3790
      N3VNT=1                                                       HARM3800
      NTVN3=NT/N3                                                   HARM3810
      MINN3=N3                                                      HARM3820
  380 JJD3 = NTSQ/N3                                                HARM3830
      M2MT=M2-MT                                                    HARM3840
  450 IF (M2MT)470,460,460                                          HARM3850
C                                                                   HARM3860
C     M2 GR. OR EQ. MT                                              HARM3870
  460 IGO2=1                                                        HARM3880
      N2VNT=N2/NT                                                   HARM3890
      MINN2=NT                                                      HARM3900
      GO TO 480                                                     HARM3910
C                                                                   HARM3920
C     M2 LESS THAN MT                                               HARM3930
  470 IGO2 = 2                                                      HARM3940
      N2VNT=1                                                       HARM3950
      NTVN2=NT/N2                                                   HARM3960
      MINN2=N2                                                      HARM3970
  480 JJD2=NTSQ/N2                                                  HARM3980
      M1MT=M1-MT                                                    HARM3990
  550 IF(M1MT)570,560,560                                           HARM4000
C                                                                   HARM4010
C     M1 GR. OR EQ. MT                                              HARM4020
  560 IGO1=1                                                        HARM4030
      N1VNT=N1/NT                                                   HARM4040
      MINN1=NT                                                      HARM4050
      GO TO 580                                                     HARM4060
C                                                                   HARM4070
C     M1 LESS THAN MT                                               HARM4080
  570 IGO1=2                                                        HARM4090
      N1VNT=1                                                       HARM4100
      NTVN1=NT/N1                                                   HARM4110
      MINN1=N1                                                      HARM4120
  580 JJD1=NTSQ/N1                                                  HARM4130
  600 JJ3=1                                                         HARM4140
      J=1                                                           HARM4150
      DO 880 JPP3=1,N3VNT                                           HARM4160
      IPP3=INV(JJ3)                                                 HARM4170
```

```
      DO 870 JP3=1,MINN3                                                HARM4180
      GO TO (610,620),IGO3                                              HARM4190
  610 IP3=INV(JP3)*N3VNT                                                HARM4200
      GO TO 630                                                         HARM4210
  620 IP3=INV(JP3)/NTVN3                                                HARM4220
  630 I3=(IPP3+IP3)*N2                                                  HARM4230
  700 JJ2=1                                                             HARM4240
      DO 870 JPP2=1,N2VNT                                               HARM4250
      IPP2=INV(JJ2)+I3                                                  HARM4260
      DO 860 JP2=1,MINN2                                                HARM4270
      GO TO (710,720),IGO2                                              HARM4280
  710 IP2=INV(JP2)*N2VNT                                                HARM4290
      GO TO 730                                                         HARM4300
  720 IP2=INV(JP2)/NTVN2                                                HARM4310
  730 I2=(IPP2+IP2)*N1                                                  HARM4320
  800 JJ1=1                                                             HARM4330
      DO 860 JPP1=1,N1VNT                                               HARM4340
      IPP1=INV(JJ1)+I2                                                  HARM4350
      DO 850 JP1=1,MINN1                                                HARM4360
      GO TO (810,820),IGO1                                              HARM4370
  810 IP1=INV(JP1)*N1VNT                                                HARM4380
      GO TO 830                                                         HARM4390
  820 IP1=INV(JP1)/NTVN1                                                HARM4400
  830 I=2*(IPP1+IP1)+1                                                  HARM4410
      IF (J-I) 840,845,845                                              HARM4420
  840 T=A(I)                                                            HARM4430
      A(I)=A(J)                                                         HARM4440
      A(J)=T                                                            HARM4450
      T=A(I+1)                                                          HARM4460
      A(I+1)=A(J+1)                                                     HARM4470
      A(J+1)=T                                                          HARM4480
  845 CONTINUE                                                          HARM4490
  850 J=J+2                                                             HARM4500
  860 JJ1=JJ1+JJD1                                                      HARM4510
C     END OF JPP1 AND JP2                                               HARM4520
C                                                                       HARM4530
  870 JJ2=JJ2+JJD2                                                      HARM4540
C     END OF JPP2 AND JP3 LOOPS                                         HARM4550
C                                                                       HARM4560
  880 JJ3 = JJ3+JJD3                                                    HARM4570
C     END OF JPP3 LOOP                                                  HARM4580
C                                                                       HARM4590
  890 IF(IFSET)891,895,895                                              HARM4600
  891 DO 892 I = 1,NX                                                   HARM4610
  892 A(2*I) = -A(2*I)                                                  HARM4620
  895 RETURN                                                            HARM4630
C                                                                       HARM4640
C     THE FOLLOWING PROGRAM COMPUTES THE SIN AND INV TABLES.            HARM4650
```

```
C                                                                       HARM4660
  900 MT=MAXO(M(1),M(2),M(3)) -2                                        HARM4670
      MT = MAXO(2,MT)                                                   HARM4680
  904 IF (MT-20)906,906,905                                             HARM4690
  905 IFERR = 1                                                         HARM4700
      GO TO 895                                                         HARM4710
  906 IFERR=0                                                           HARM4720
      NT=2**MT                                                          HARM4730
      NTV2=NT/2                                                         HARM4740
C                                                                       HARM4750
CC    SET UP SIN TABLE                                                  HARM4760
C     THETA=PIE/2**(L+1) FOR L=1                                        HARM4770
  910 THETA=.7853981634                                                 HARM4780
C                                                                       HARM4790
C     JSTEP=2**(MT-L+1) FOR L=1                                         HARM4800
      JSTEP=NT                                                          HARM4810
C                                                                       HARM4820
CC    JDIF=2**(MT-L) FOR L=1                                            HARM4830
C     JDIF=2**(MT-L) FOR L=1                                            HARM4840
      JDIF=NTV2                                                         HARM4850
      S(JDIF)=SIN(THETA)                                                HARM4860
      DO 950 L=2,MT                                                     HARM4870
      THETA=THETA/2.                                                    HARM4880
      JSTEP2=JSTEP                                                      HARM4890
      JSTEP=JDIF                                                        HARM4900
      JDIF=JSTEP/2                                                      HARM4910
      S(JDIF)=SIN(THETA)                                                HARM4920
      JC1=NT-JDIF                                                       HARM4930
      S(JC1)=COS(THETA)                                                 HARM4940
      JLAST=NT-JSTEP2                                                   HARM4950
      IF(JLAST - JSTEP) 950,920,920                                     HARM4960
  920 DO 940 J=JSTEP,JLAST,JSTEP                                        HARM4970
      JC=NT-J                                                           HARM4980
      JD=J+JDIF                                                         HARM4990
  940 S(JD)=S(J)*S(JC1)+S(JDIF)*S(JC)                                   HARM5000
  950 CONTINUE                                                          HARM5010
C                                                                       HARM5020
CC    SET UP INV(J) TABLE                                               HARM5030
C                                                                       HARM5040
  960 MTLEXP=NTV2                                                       HARM5050
C                                                                       HARM5060
C     MTLEXP=2**(MT-L). FOR L=1                                         HARM5070
      LM1EXP=1                                                          HARM5080
C                                                                       HARM5090
C     LM1EXP=2**(L-1). FOR L=1                                          HARM5100
      INV(1)=0                                                          HARM5110
      DO 980 L=1,MT                                                     HARM5120
      INV(LM1EXP+1) = MTLEXP                                            HARM5130
```

```
      DO 970 J=2,LM1EXP                                                 HARM5140
      JJ=J+LM1EXP                                                       HARM5150
  970 INV(JJ)=INV(J)+MTLEXP                                             HARM5160
      MTLEXP=MTLEXP/2                                                   HARM5170
  980 LM1EXP=LM1EXP*2                                                   HARM5180
  982 IF(IFSET)12,895,12                                                HARM5190
      END                                                               HARM5200



      SUBROUTINE DERFUN(Y,DY,T)                                         DERFU000
      DIMENSION Y(1),DY(1)                                              DERFU001
      DATA C/100./
      DY(1)=0.
      DO 5 I=2,200
      DY(I)=-C *(Y(I+1)-Y(I-1))
    5 CONTINUE                                                          DERFU006
      DY(201)=DY(200)
      RETURN                                                            DERFU011
      END                                                               DERFU012
      DOUBLE PRECISION UY                                               MAIN  000
      DIMENSION UY(1005),VF(1005),YD(808),Y(201),DY(201)
      CALL RKAM(Y,DY,YD,VF,UY)                                          MAIN  002
      END                                                               MAIN  003



      SUBROUTINE RKAM(Y,DY,YD,VF,UY)                                    RKAM  000
      DOUBLE PRECISION UY                                               RKAM  001
      DIMENSION Y(1),UY(1),VF(1),YD(1),A(20),D(20),ID(20),B(20),DY(1)   RKAM  002
     *,XV(5),BET(4),IOR(10),IOW(10),JXY(10),YX(101,6),MYY(6),YY(101,6), RKAM  003
     *IYX(4),IYY(4)                                                     RKAM  004
      DIMENSION XX(101),YYG(101),LAB(1)
      EQUIVALENCE (M1,NN),(M2,MODE),(A2,ERMAX),(A3,ERMIN),(A4,STMAX),
     *(A5,STMIN),(A6,FACT)
      EQUIVALENCE (XX(1),YX(1,1)),(YYG(1),YX(1,2))
      REAL IT(24)/96HP(X,T) VS. X FOR T=          SEC. DX=0.01       R./
     *JOHNSON BOX 70
      DEFINE FILE 8(5,100,E,J9)
      DATA LA/4H     /
      DATA IOR/4H(8E1,4H0.3),8*4H    /,IOW/4H(1H+,4H,11X,4H6(I4,4H,1PE,4
     *H12.4,4H)/(1,4H2X,6,4H(I4,,4HE12.,4H4))/
      DATA BET/0.5,0.5,1.0,0.0/,MYY/1,2,3,4,5,6/,JXY/1,2,1,3,1,4,1,5,1, RKAM  007
     *6/,IYP/3H Y(/,MP/3H Y(/                                           RKAM  008
C RUNGE-KUTTA ADAMS MOULTON SOLUTION OF N SIMULTANEOUS FIRST ORDER      RKAM  009
C ORDINARY DIFFERENTIAL EQUATIONS USING PARTIAL DOUBLE PRECISION ARITH- RKAM  010
```

```
C METIC                                                                 RKAM 011
C                                                                       RKAM 012
C  A1=THE INITIAL VALUE OF THE STEP SIZE*********************************RKAM 013
C  A2=THE MAXIMUM ABSOLUTE VALUE OF THE ALLOWABLE SINGLE STEP ERROR******RKAM 014
C  A3= THE MINIMUM ABSOLUTE VALUE OF THE ALLOWABLE SINGLE STEP ERROR*****RKAM 015
C  A4=THE MAXIMUM ABSOLUTE VALUE OF STEP SIZE****************************RKAM 016
C  A5=THE MINIMUMM ABSOLUTE VALUE OF STEP SIZE***************************RKAM 017
C  A6=REDUCTION FACTOR FOR THE STEP SIZE*********************************RKAM 018
C  M1=NUMBER OF FIRST ORDER EQUATIONS************************************RKAM 019
C  M2=MODE OF INTEGRATION, M2=1=RK,M2=2=RKAM,M2=3=RKAM WITH ERROR CHECK*RKAM 020
C  M3=TYPE OF ERROR CHECK,M3=1=REL. ERROR,M3=2=ABSOLUTE ERROR***********RKAM 021
C  M4=PRINT OR SUPPRESS ERROR MESSAGES,M4=1=PRINT MESSAGES,M4=2=DELETE   RKAM 022
C  MESSAGES**************************************************************RKAM 023
C  M5=PRINTING INTERVAL,M5>0=PRINT ACCORDING TO SPECIFICATIONS,M5=0=      RKAM 024
C  PRINT AT EACH INTEGRATION STEP****************************************RKAM 025
C  M6=FORMAT OPTION,IF M6=0 USE INTERNAL FORMATS;IF M6>0 READ NEW FORMAT
C                                                                       RKAM 026
C IDENTIFICATION DATA-BLANK CARD WILL SUPPRESS*****************************RKAM 027
C                                                                       RKAM 028
      READ(5,30)(ID(I),I=1,20)                                          RKAM 029
C                                                                       RKAM 030
C READ CONTROL PARAMETERS*************************************************
C
      READ(5,20)M1,M2,M3,M4,M5,M6,IYVSX,IYVSY
C
C IOR=INPUT READ FORMAT,IOW=OUTPUT WRITE FORMAT****************************RKAM 031
C
      IF(M6)2,3,2
    2 READ(5,30)(IOR(I),I=1,10),(IOW(J),J=1,10)                         RKAM 033
C                                                                       RKAM 032
    3 READ(5,IOR)A1,A2,A3,A4,A5,A6                                      RKAM 035
C                                                                       RKAM 036
C INITIAL AND FINAL VALUE OF THE INDEPENDENT VARIABLE*********************RKAM 037
C                                                                       RKAM 038
      READ(5,IOR)ALPHA,OMEGA                                            RKAM 039
C                                                                       RKAM 040
C INITIAL VALUES OF THE DEPENDENT VARIABLES*******************************RKAM 041
      READ(5,IOR)(Y(I),I=1,M1)                                          RKAM 042
C                                                                       RKAM 043
      LT=2                                                              RKAM 044
      IF(M5)121,121,119                                                 RKAM 045
  119 READ(5,IOR)(A(I),D(I),B(I),I=1,M5)                                RKAM 046
      LT=1                                                              RKAM 047
  121 CONTINUE                                                          RKAM 048
      WRITE(6,220)                                                      RKAM 049
      WRITE(6,10)(ID(I),I=1,20)                                         RKAM 050
      WRITE(6,70)                                                       RKAM 054
      WRITE(6,80)M1,M2,M3,M4,M5,M6,IYVSX,IYVSY
```

```
      WRITE(6,90)                                                       RKAM 052
      WRITE(6,120)A1,A2,A3,A4,A5,A6                                     RKAM 053
      WRITE(6,60)ALPHA,OMEGA                                            RKAM 056
      WRITE(6,110)                                                      RKAM 057
      WRITE(6,IOW)(I,Y(I),I=1,M1)                                       RKAM 058
      WRITE(6,40)(IOR(I),I=1,10),(IOW(J),J=1,10)                        RKAM 051
C
CC GRAPH READ STATEMENTS*************************************************
C
      IF(IYVSX)335,335,331                                              RKAM 059
  331 READ(5,20)NYX,(IYX(I),I=1,NYX)                                    RKAM 060
      WRITE(6,50)NYX,(IYP,IYX(I),I=1,NYX)                               RKAM 061
  335 IF(IYVSY)337,337,336                                              RKAM 062
  336 READ(5,20)NYY,(IYY(2*I-1),IYY(2*I),I=1,NYY)                       RKAM 063
      WRITE(6,150)(MP,IYY(2*I-1),IYY(2*I),I=1,NYY)                      RKAM 064
  337 CONTINUE                                                          RKAM 065
      LL=1                                                              RKAM 066
      KM=1                                                              RKAM 067
      INP=0                                                             RKAM 068
      IND=0                                                             RKAM 069
      JNP=0                                                             RKAM 070
      JND=0                                                             RKAM 071
      T=ALPHA                                                           RKAM 072
      WRITE(6,220)                                                      RKAM 073
C                                                                       RKAM 074
CC SET CONSTANTS ON RUNGE-KUTTA AND (IF USED), ADAMS-MCULTON************RKAM 075
C                                                                       RKAM 076
      F16TH=1./6.                                                       RKAM 077
      F24=1./24.                                                        RKAM 078
      GO TO (7,88,88),M2                                                RKAM 079
    7 MM=4                                                              RKAM 080
      J1=4                                                              RKAM 081
      GO TO 9                                                           RKAM 082
   88 MM=1                                                              RKAM 083
      J1=1                                                              RKAM 084
    9 EPM=2.0*ABS(OMEGA-ALPHA)                                          RKAM 085
      E3=EPM                                                            RKAM 086
      ZOT=.5+2.0**(-30.0)                                               RKAM 087
      N2=NN+2                                                           RKAM 088
      DT=A1                                                             RKAM 089
      R=19.0/270.0                                                      RKAM 090
      XV(MM)=T                                                          RKAM 091
      IF(ERMIN)11,11,13
   11 ERMIN=ERMAX/55.
   13 IF(FACT)14,14,15
   14 FACT=0.5
   15 CALL DERFUN(Y,DY,T)
      MMM=(MM-1)*NN                                                     RKAM 097
```

```
      DO 320 I=1,NN                                                    RKAM 098
      IMM=MMM+I                                                        RKAM 099
      VF(IMM)=DY(I)                                                    RKAM 100
  320 UY(IMM)=Y(I)                                                     RKAM 101
      GO TO (501,502),LT                                               RKAM 102
  501 E3=ABS(A(1)-B(1))                                                RKAM 103
      IF(D(1))505,504,505                                              RKAM 104
  504 E3=ABS(DT*0.5)                                                   RKAM 105
  505 ABM=E3                                                           RKAM 106
      WRITE(6,400)KM,A(KM),D(KM),B(KM)                                 RKAM 107
      WRITE(6,130)                                                     RKAM 108
      GO TO 1001                                                       RKAM 109
  502 WRITE(6,100)                                                     RKAM 110
C                                                                      RKAM 111
C RUNGE-KUTTA ROUTINE*************************************************RKAM 112
C                                                                      RKAM 113
 1001 MMM=(MM-1)*NN                                                    RKAM 114
      KI=-NN                                                           RKAM 115
      DO 1034 K=1,4                                                    RKAM 116
      KI=KI+NN                                                         RKAM 117
      DO 1350 I=1,NN                                                   RKAM 118
      IK=KI+I                                                          RKAM 119
      IMM=MMM+I                                                        RKAM 120
      YD(IK)=DT*VF(IMM)                                                RKAM 121
 135C Y(I)=UY(IMM)+BET(K)*YD(IK)                                       RKAM 122
      T=BET(K)*DT+XV(MM)                                               RKAM 123
      CALL DERFUN(Y,DY,T)                                              RKAM 124
      DO 1100 I=1,NN                                                   RKAM 125
 110C VF(I+MMM)=DY(I)                                                  RKAM 126
 1034 CONTINUE                                                         RKAM 127
      DO 1039 I=1,NN                                                   RKAM 128
      I2=I+NN                                                          RKAM 129
      I3=I2+NN                                                         RKAM 130
      I4=I3+NN                                                         RKAM 131
      IMM=MMM+I                                                        RKAM 132
      UY(IMM+NN)=UY(IMM)+(YD(I)+2.*(YD(I2)+YD(I3))+YD(I4))*F16TH       RKAM 133
 1039 CONTINUE                                                         RKAM 134
      MM=MM+1                                                          RKAM 135
      XV(MM)=XV(MM-1)+DT                                               RKAM 136
      MMM=(MM-1)*NN                                                    RKAM 137
      DO 1400 I=1,NN                                                   RKAM 138
 1400 Y(I)=UY(MMM+I)                                                   RKAM 139
      T=XV(MM)                                                         RKAM 140
      CALL DERFUN(Y,DY,T)                                              RKAM 141
      GO TO (42,99,99),MODE                                            RKAM 142
   99 MMM=(MM-1)*NN                                                    RKAM 143
      DO 151 I=1,NN                                                    RKAM 144
  151 VF(MMM+I)=DY(I)                                                  RKAM 145
```

```
      IF(MM-3)1001,1001,2000                                                    RKAM  146
C                                                                               RKAM  147
C ADAMS-MOULTON ROUTINE*****************************************************RKAM  148
C                                                                               RKAM  149
 2000 DO 2048 I=1,NN                                                            RKAM  150
      I2=I+NN                                                                   RKAM  151
      I3=I2+NN                                                                  RKAM  152
      I4=I3+NN                                                                  RKAM  153
      I5=I4+NN                                                                  RKAM  154
      Y(I)=UY(I4)+DT*(55.*VF(I4)-59.*VF(I3)+37.*VF(I2)-9.*VF(I))*F24            RKAM  155
 2048 YD(I)=Y(I)                                                                RKAM  156
      T=XV(4)+DT                                                                RKAM  157
      CALL DERFUN(Y,DY,T)                                                       RKAM  158
      XV(5)=T                                                                   RKAM  159
      DO 2051 I=1,NN                                                            RKAM  160
      I2=I+NN                                                                   RKAM  161
      I3=I2+NN                                                                  RKAM  162
      I4=I3+NN                                                                  RKAM  163
      I5=I4+NN                                                                  RKAM  164
      UY(I5)=UY(I4)+DT*(9.*DY(I)+19.*VF(I4)-5.*VF(I3)+VF(I2))*F24               RKAM  165
 2051 Y(I)=UY(I5)                                                               RKAM  166
      CALL DERFUN(Y,DY,T)                                                       RKAM  167
      GO TO (42,42,3000),MODE                                                   RKAM  168
C                                                                               RKAM  169
C ERROR CHECKING ROUTINE*****************************************************QRKAM  170
C                                                                               RKAM  171
 3000 SSE=0.0                                                                   RKAM  172
      DO 3033 I=1,NN                                                            RKAM  173
      EPSIL=R*ABS(Y(I)-YD(I))                                                   RKAM  174
      GO TO (3301,3307),M3                                                      RKAM  175
 3301 IF(Y(I))3650,3307,3650                                                    RKAM  176
 3650 EPSIL=EPSIL/(ABS(Y(I)))                                                   RKAM  177
 3307 IF(SSE-EPSIL)3032,3033,3033                                               RKAM  178
 3032 SSE=EPSIL                                                                 RKAM  179
 3033 CONTINUE                                                                  RKAM  180
      IF(ERMAX-SSE)3034,3034,3035                                               RKAM  181
 3034 IF(ABS(DT)-STMIN)42,42,4340                                               RKAM  182
 3035 IF(SSE-ERMIN)3036,42,42                                                   RKAM  183
 3036 IF(STMAX-ABS(DT))42,42,5360                                               RKAM  184
 4340 LL=1                                                                      RKAM  185
      MM=1                                                                      RKAM  186
      GO TO (4560,4660),M4                                                      RKAM  187
 4560 WRITE(6,800)SSE,T,DT                                                      RKAM  188
      DT=DT*FACT                                                                RKAM  189
      WRITE(6,820)XV(1),DT                                                      RKAM  190
      GO TO 1001                                                                RKAM  191
 4660 DT=DT*FACT                                                                RKAM  192
      GO TO 1001                                                                RKAM  193
```

```
 5360 GO TO (42,5361),LL                                                RKAM  194
 5361 XV(2)=XV(3)                                                       RKAM  195
      XV(3)=XV(5)                                                       RKAM  196
      DO 5363 I=1,NN                                                    RKAM  197
      I2=I+NN                                                           RKAM  198
      I3=I2+NN                                                          RKAM  199
      I5=I3+I3-I                                                        RKAM  200
      VF(I2)=VF(I3)                                                     RKAM  201
      VF(I3)=DY(I)                                                      RKAM  202
      UY(I2)=UY(I3)                                                     RKAM  203
 5363 UY(I3)=UY(I5)                                                     RKAM  204
      LL=2                                                              RKAM  205
      MM=3                                                              RKAM  206
      GO TO (5270,5370),M4                                              RKAM  207
 5270 WRITE(6,800)SSE,T,DT                                              RKAM  208
      DT=DT*2.                                                          RKAM  209
      WRITE(6,840)DT                                                    RKAM  210
      WRITE(6,130)                                                      RKAM  211
      GO TO 1001                                                        RKAM  212
 5370 DT=2.*DT                                                          RKAM  213
      GO TO 1001                                                        RKAM  214
C                                                                       RKAM  215
C EXIT ROUTINE**********************************************************RKAM  216
C                                                                       RKAM  217
   42 II5=4*NN                                                          RKAM  218
      DO 322 I=1,NN                                                     RKAM  219
  322 VF(II5+I)=DY(I)                                                   RKAM  220
      GO TO (701,602),LT                                                RKAM  221
  142 KI=-NN                                                            RKAM  222
      DO 12 K=1,3                                                       RKAM  223
      KI=KI+NN                                                          RKAM  224
      XV(K)=XV(K+1)                                                     RKAM  225
      DO 12 I=1,NN                                                      RKAM  226
      IK=KI+I                                                           RKAM  227
      VF(IK)=VF(IK+NN)                                                  RKAM  228
   12 UY(IK)=UY(IK+NN)                                                  RKAM  229
      LL=2                                                              RKAM  230
      MM=4                                                              RKAM  231
      XV(4)=XV(5)                                                       RKAM  232
      II4=3*NN                                                          RKAM  233
      DO 52 I=1,NN                                                      RKAM  234
      I4=II4+I                                                          RKAM  235
      VF(I4)=VF(I4+NN)                                                  RKAM  236
   52 UY(I4)=UY(I4+NN)                                                  RKAM  237
      GO TO (1001,2000,2000),MODE                                       RKAM  238
C                                                                       RKAM  239
C WRITE OUTPUT WITH SPECIFICATIONS**************************************RKAM  240
C                                                                       RKAM  241
```

```
  701 TZ=ABS(DT)*ZCT                                                    RKAM  242
      GO TO (700,750,750),MODE                                          RKAM  243
  700 KT=4                                                              RKAM  244
      GO TO 430                                                         RKAM  245
  750 KT=1                                                              RKAM  246
  430 SPACE=A(KM)-XV(KT)                                                RKAM  247
      Z=ABS(SPACE)                                                      RKAM  248
      IF(Z-TZ)437,437,436                                               RKAM  249
  437 IF(SPACE)413,403,413                                              RKAM  250
  403 KTM=(KT-1)*NN                                                     RKAM  251
      DO 404 I=1,NN                                                     RKAM  252
  404 Y(I)=UY(KTM+I)                                                    RKAM  253
      GO TO 443                                                         RKAM  254
  436 KT=KT+1                                                           RKAM  255
      IF(KT-5)430,430,790                                               RKAM  256
  413 KTM=(KT-1)*NN                                                     RKAM  257
      DO 438 I=1,NN                                                     RKAM  258
  438 DY(I)=VF(I+KTM)                                                   RKAM  259
      KI=-NN                                                            RKAM  260
      DO 439 K=1,4                                                      RKAM  261
      KI=KI+NN                                                          RKAM  262
      DO 440 I=1,NN                                                     RKAM  263
      IK=KI+I                                                           RKAM  264
      YD(IK)=SPACE*DY(I)                                                RKAM  265
  440 Y(I)=UY(I+KTM)+BET(K)*YD(IK)                                      RKAM  266
      T=BET(K)*SPACE+XV(KT)                                             RKAM  267
  439 CALL DERFUN(Y,DY,T)                                               RKAM  268
      DO 442 I=1,NN                                                     RKAM  269
      I2=I+NN                                                           RKAM  270
      I3=I2+NN                                                          RKAM  271
      I4=I3+NN                                                          RKAM  272
  442 Y(I)=UY(KTM+I)+(YD(I)+2.*(YD(I2)+YD(I3))+YD(I4))*F16TH             RKAM  273
  443 WRITE(6,140)A(KM)                                                 RKAM  274
      WRITE(6,IOW)(I,Y(I),I=1,NN)                                       RKAM  275
C                                                                       RKAM  276
      U=Y(1)*Y(1)*0.01                                                  RKAM  285
      DO 948 I=2,201
      U=U+0.01*Y(I)*Y(I)
  948 CONTINUE                                                          RKAM  288
      WRITE(6,860)U
  860 FORMAT(3H U=,1PE10.3)
      DX=0.01
      X=0.                                                              RKAM  278
      DO 947 I=1,101
      YX(I,1)=X                                                         RKAM  280
      X=X+DX                                                            RKAM  281
      YX(I,2)=Y(2*I-1)                                                  RKAM  282
  947 CONTINUE                                                          RKAM  283
```

```
      WRITE(8'1,500)T
  500 FORMAT(F8.2)
      READ(8'1,510)IT(6),IT(7)
  510 FORMAT(2A4)
      CALL VPLOT(YX,JXY,101,101,1,0,0.0,0.0,0.0,0.0)
      CALL DRAW(101,XX,YYG,0,0,LA,IT,.2,1.,2,0,2,2,5,4,1,LAST)
C GRAPH ROUTINE*************************************************************RKAM  291
C                                                                             RKAM  292
      IF(IYVSX)471,471,451                                                    RKAM  293
  451 INP=INP+1                                                               RKAM  294
      IF(INP-IYVSX)471,455,471                                                RKAM  295
  455 INP=0                                                                   RKAM  296
      IND=IND+1                                                               RKAM  297
      YX(IND,1)=A(KM)                                                         RKAM  298
      DO 457 I=1,NYX                                                          RKAM  299
  457 YX(IND,I+1)=Y(IYX(I))                                                   RKAM  300
      IF(IND-100)471,459,471                                                  RKAM  301
  459 IND=0                                                                   RKAM  302
      CALL VPLOT(YX,JXY,100,100,NYX,0,0.0,0.0,0.0,0.0)                        RKAM  303
      IF(JND+1-100)461,471,461                                                RKAM  304
  461 WRITE(6,220)                                                            RKAM  305
      WRITE(6,130)                                                            RKAM  306
  471 IF(IYVSY)491,491,473                                                    RKAM  307
  473 JNP=JNP+1                                                               RKAM  308
      IF(JNP-IYVSY)491,475,491                                                RKAM  309
  475 JNP=0                                                                   RKAM  310
      JND=JND+1                                                               RKAM  311
      I2=0                                                                    RKAM  312
      DO 477 I=1,NYY                                                          RKAM  313
      I2=I2+2                                                                 RKAM  314
      YY(JND,I2-1)=Y(IYY(I2-1))                                               RKAM  315
  477 YY(JND,I2)=Y(IYY(I2))                                                   RKAM  316
      IF(JND-100)491,479,491                                                  RKAM  317
  479 CALL VPLOT(YY,MYY,100,100,NYY,0,0.0,0.0,0.0,0.0)                        RKAM  318
      JND=0                                                                   RKAM  319
      WRITE(6,220)                                                            RKAM  320
      WRITE(6,130)                                                            RKAM  321
  491 CONTINUE                                                                RKAM  322
  733 IF(D(KM))715,746,715                                                    RKAM  323
  715 A(KM)=A(KM)+D(KM)                                                       RKAM  324
      E=ABS(A(KM)-B(KM))                                                      RKAM  325
      IF(E-ABM)744,746,746                                                    RKAM  326
  744 ABM=E                                                                   RKAM  327
      GO TO 430                                                               RKAM  328
  746 KM=KM+1                                                                 RKAM  329
      IF(KM-M5)702,702,648                                                    RKAM  330
  702 WRITE(6,220)                                                            RKAM  331
      WRITE(6,400)KM,A(KM),D(KM),B(KM)                                        RKAM  332
```

```
      E3=ABS(A(KM)-B(KM))                                                    RKAM  333
      IF(D(KM))714,713,714                                                   RKAM  334
  713 E3=ABS(DT*0.5)                                                         RKAM  335
  714 ABM=E3                                                                 RKAM  336
      GO TO (700,750,750),MODE                                               RKAM  337
  790 E=ABS(XV(J1)-OMEGA)                                                    RKAM  338
      IF(E-EPM)706,648,648                                                   RKAM  339
  706 EPM=E                                                                  RKAM  340
      GO TO 142                                                              RKAM  341
C                                                                            RKAM  342
CC  WRITE OUTPUT AT EACH MESHPOINT*****************************************RKAM  343
C                                                                            RKAM  344
  602 E=ABS(OMEGA-XV(J1))                                                    RKAM  345
      IF(E-EPM)672,672,648                                                   RKAM  346
  672 EPM=E                                                                  RKAM  347
      GO TO (600,650,650),MODE                                               RKAM  348
  600 KT=4                                                                   RKAM  349
      GO TO 630                                                              RKAM  350
  650 KT=1                                                                   RKAM  351
  630 E=ABS(OMEGA-XV(KT))                                                    RKAM  352
      IF(E-E3)647,645,645                                                    RKAM  353
  647 E3=E                                                                   RKAM  354
      KTM=(K-1)*NN                                                           RKAM  355
      DO 666 I=1,NN                                                          RKAM  356
  666 Y(I)=UY(KTM+I)                                                         RKAM  357
      CALL DERFUN(Y,DY,T)                                                    RKAM  358
      WRITE(6,140)XV(KT)                                                     RKAM  359
      WRITE(6,100)(I,Y(I),I=1,NN)                                            RKAM  360
      IF(LT-1)733,733,645                                                    RKAM  361
  645 KT=KT+1                                                                RKAM  362
      IF(KT-5)630,630,142                                                    RKAM  363
  648 IF(IYVSX)651,651,649                                                   RKAM  364
  649 CALL VPLOT(YX,JXY,IND,100,NYX,0,0.0,0.0,0.0,0.0,0.0)                   RKAM  365
  651 IF(IYVSY)655,655,653                                                   RKAM  366
  653 CALL VPLOT(YY,MYY,JND,100,NYY,0,0.0,0.0,0.0,0.0,0.0)                   RKAM  367
  655 WRITE(6,220)                                                           RKAM  368
      RETURN                                                                 RKAM  369
   10 FORMAT(1X,20A4)                                                        RKAM  370
   20 FORMAT(16I5)                                                           RKAM  371
   30 FORMAT(20A4)                                                           RKAM  372
   40 FORMAT(//21H INPUT-OUTPUT FORMATS/1X,10A4,10A4)                        RKAM
   50 FORMAT(17H PRINTER PLOT OF ,I2,38H VARIABLE(S) VS. INDEPENDENT VARRKAM  374
     *IABLE./22H VARIABLES PLOTTED ARE,3(A3,I2,1H)))                         RKAM  375
   60 FORMAT(/9H ALPHA = ,1PE12.5,10X,9H OMEGA = ,E12.5/)                    RKAM  376
   70 FORMAT(/48H     M1     M2     M3     M4     M5     M6 IYVSX IYVSY)
   80 FORMAT(16I6)                                                           RKAM  378
   90 FORMAT(71H     INITIAL    MAX. ALLOW. MIN. ALLOW. MAX. ALLOW. MIN. ARKAM  379
     *LLOW. STEP SIZE/72H      STEP      SINGLE STEP SINGLE STEP  ABSOLUTERKAM  380
```

```
     *     ABSOLUTE    REDUCTION/69H     SIZE          ERROR         ERROR    RKAM 381
     * STEP SIZE     STEP SIZE    FACTOR/)                                     RKAM 382
  100 FORMAT(40X,28H WRITEING AT EACH MESH POINT//)                            RKAM 383
  110 FORMAT(39H STARTING VALUES OF DEPENDENT VARIABLES//)                     RKAM 384
  120 FORMAT(1P6E12.4)                                                         RKAM 385
  130 FORMAT(12H INDEPENDENT,8X,32HDEPENDENT VARIABLE NO. AND VALUE/           RKAM 386
     *10H  VARIABLE/)                                                          RKAM 387
  140 FORMAT(1P1E12.4)                                                         RKAM 388
  150 FORMAT(23H PRINTER PHASE PLOT OF ,2(A3,I2,8H) VS. Y(,I2,1H)))            RKAM 389
  220 FORMAT(1H1)                                                              RKAM 390
  400 FORMAT(30X,18HWRITEING INTERVAL,I3//10H START AT ,1PE14.6,5X, 17H RKAM 391
     *INCREMENTING BY ,1E14.6,5X,15H AND FINISH AT ,1E14.6//)                  RKAM 392
  800 FORMAT(/18H MAXIMUM ERROR OF ,1P1E12.5,4H AT ,1P1E12.5,19H WITH STRKAM 393
     *EP SIZE OF ,1P1E12.5)                                                    RKAM 394
  820 FORMAT(  27H ERROR TOO LARGE, BEGIN AT ,1P1E12.5,19H WITH STEP SIZRKAM 395
     *E OF ,1P1E12.5/)                                                         RKAM 396
  840 FORMAT( 32H ERROR TOO SMALL. NEW STEP SIZE ,1PE12.5/)                    RKAM 397
      END                                                                      RKAM 398


      SUBROUTINE VPLOT(XY,JXY,N,NDIM,NCUR,ISCALE,XL,XU,YL,YU)                  000
      DIMENSION IGRID(101),XS(11),YS(17),ICHAR(7),XY(1),JXY(1)                 001
      DATA ICHAR/1H+,1H*,1H¬,1H$,1H=,1H.,1H /                                  002
      XMIN=XL                                                                  003
      XMAX=XU                                                                  004
      YMIN=YL                                                                  005
      YMAX=YU                                                                  006
      IF(ISCALE.NE.0)GO TO 32                                                  007
      XMAX=-1.0E 20                                                            008
      XMIN=-XMAX                                                               009
      YMAX=XMAX                                                                010
      YMIN=XMIN                                                                011
      J2=0                                                                     012
      DO 31 J=1,NCUR                                                           013
      J2=J2+2                                                                  014
      JIX=(JXY(J2-1)-1)*NDIM                                                   015
      JIY=(JXY(J2)-1)*NDIM                                                     016
      DO 31 I=1,N                                                              017
      IJX=JIX+I                                                                018
      IJY=JIY+I                                                                019
      IF(XY(IJX).GT.XMAX)XMAX=XY(IJX)                                          020
      IF(XY(IJX).LT.XMIN)XMIN=XY(IJX)                                          021
      IF(XY(IJY).GT.YMAX)YMAX=XY(IJY)                                          022
      IF(XY(IJY).LT.YMIN)YMIN=XY(IJY)                                          023
   31 CONTINUE                                                                 024
   32 XR=XMAX-XMIN                                                             025
```

```
      IF(XR.EQ.C.0)XR=1.0E-20                                          026
      YR=YMAX-YMIN                                                     027
      IF(YR.EQ.0.0)YR=1.0E-20                                          028
      XT=XMAX*XMIN                                                     029
      YT=YMIN*YMAX                                                     030
      IF(XT.LT.0.0)IYAX=100.0*(-XMIN)/XR+1.5                           031
      IF(YT.LE.0.0)IXAX=64.0*YMAX/YR+1.5                               032
      XINCR=XR/10.C                                                    033
      YINCR=YR/16.C                                                    034
      XS(1)=XMIN                                                       035
      YS(1)=YMAX                                                       036
      DO 46 I=2,11                                                     037
   46 XS(I)=XS(I-1)+XINCR                                              038
      DO 47 I=2,17                                                     039
   47 YS(I)=YS(I-1)-YINCR                                              040
      WRITE(6,10)(XS(I),I=1,11)                                        041
      II=1                                                             042
      KK=0                                                             043
      DO 146 LINE=1,65                                                 044
      DO 101 J=1,101                                                   045
  101 IGRID(J)=ICHAR(7)                                                046
      IF(YT.GT.0.0)GO TO 109                                           047
      IF(LINE.NE.IXAX)GO TO 109                                        048
      DO 105 J=1,101                                                   049
  105 IGRID(J)=ICHAR(6)                                                050
  109 IF(XT.LT.C.0)IGRID(IYAX)=ICHAR(6)                                051
      J2=0                                                             052
      DO 125 J=1,NCUR                                                  053
      JC=MOD(J,5)                                                      054
      J2=J2+2                                                          055
      JIX=(JXY(J2-1)-1)*NDIM                                           056
      JIY=(JXY(J2)-1)*NDIM                                             057
      DO 125 I=1,N                                                     058
      IJX=JIX+I                                                        059
      IJY=JIY+I                                                        060
      IPTY=64.C*(YMAX-XY(IJY))/YR+1.5                                  061
      IF(IPTY.GT.65)IPTY=65
      IF(IPTY.LT.1)IPTY=1                                              063
      IF(IPTY.NE.LINE)GO TO 125                                        064
      IPTX=10C.0*(XY(IJX)-XMIN)/XR+1.5                                 065
      IF(IPTX.LT.1)IPTX=1                                              066
      IF(IPTX.GT.101)IPTX=101
      IF(JC.NE.0)GC TO 119                                             068
      IGRID(IPTX)=ICHAR(5)                                             069
      GO TO 125                                                        070
  119 IGRID(IPTX)=ICHAR(JC)                                            071
  125 CONTINUE                                                         072
      IF(KK.GT.0)GC TO 134                                             073
```

```
      WRITE(6,20)YS(II),(IGRID(I),I=1,101),YS(II)                       074
      II=II+1                                                           075
      GO TO 135                                                         076
  134 WRITE(6,30)(IGRID(I),I=1,101)                                     077
  135 KK=KK+1                                                           078
      IF(KK.NE.4)GO TO 146                                              079
      KK=0                                                              080
  146 CONTINUE                                                          081
      WRITE(6,40)(XS(I),I=1,11)                                         082
      RETURN                                                            083
   10 FORMAT(1H1,1PE15.2,10E10.2/10X,1H*,20(5H+****),2H+*)              084
   20 FORMAT(1PE10.2,1H+,101A1,1H+,E9.2)                                085
   30 FORMAT(10X,1H*,101A1,1H*)                                         086
   40 FORMAT(10X,1H*,20(5H+****),2H+*/1PE16.2,10E10.2)                  087
      END                                                               088


      BLOCK DATA
C-------------------------------------------------------------
CCC         THIS PROGRAM IS A SIMULATION OF THE 'WIDROW'
CCC       FILTER USING THE IBM 2250 GRAPHIC DISPLAY UNIT.
CCC       NO INPUT DATA CARDS ARE REQUIRED. FILTER PARAMETERS
CCC       MAY BE CHANGED FROM THE 2250. THE ONLY INPUT THAT
CCC       IS REQUIRED IS A 9-TRACK,STANDARD LABEL,800 BPI TAPE
CCC       ON WHICH IS THE ACTUAL OR SIMULATED SIGNAL IN DIGITAL
CCC       FORM WHICH WOULD APPEAR AT THE OUTPUT OF THE ARRAY
CCC       SENSORS.
C-------------------------------------------------------------
C-------------------------------------------------------------
CCC
CCC          BLOCK DATA ESTABLISHES NOMINAL INITIAL VALUES
CCC       FOR VARIABLES COMMON TO THE MAIN PROGRAM AND THE
CCC       SUBROUTINES.  NOMINAL VALUES FOR MOST INPUT PARA-
CCC       METERS ARE LATER DISPLAYED ON THE 2250 CRT DISPLAY
CCC       UNIT AND REPLACED WITH THE DESIRED VALUES FOR THE
CCC       NUMERIC KEYBOARD.
CCC
C-------------------------------------------------------------
      COMMON/VALUES/A,D,X(11,48),W(11,11),DESIG,SIGOUT,SIG,
     *DT,SPACE,FO,BW,C,DF,KS,ANG,T1,IT1,IT2,IT3,IT4,IT5,LAB,
     *ANGL,FREQ,XP(128),XP2(505),YP2(505),TEMP(15),S(64),
     *ANGOLD,XY(256,4),JXY(4)
      COMMON/SIZES/NE,ND,NIT,NCPLX,NF,NA,NT,NN,KDEL,INV(64),
     *IT,N1,NNJ,ICODE2,MP,NP,NA1,NF1,NULL(1)
      COMMON/IPLOTS/IPLOT1,IPLOT2,IPLOT3,IPLOT4,IATL,ITEXT1,
     *ITEXT2,IATN1,IKEY1
```

```
      COMMON/SCALES/XMIN,XMAX,YMAX,FMIN,FMAX,YMIN,ZERO,TEN
      COMPLEX A(256),D(256)
C
CC        ENTER NOMINAL INITIAL VALUES FOR VARIABLES.
C
C  NE=NUMBER OF ELEMENTS,
C  ND=NUMBER OF DELAY ELEMENTS
C  NIT=NUMBER OF ITTERATICNS OF THE FILTER
C  NT=NUMBER OF SAMPLING TIME DELAYS PER DELAY ELEMENT
C  FO=SIGNAL CENTER FREQUENCY
C  BW=SIGNAL BANDWIDTH OF DESIRED SIGNAL
C  DT=SAMPLING TIME INTERVAL
      REAL KS/0.0001/,FO/500./,BW/200./,C/1500./,SPACE/1.5/,
     *T1/'  CO'/,ANG/0.0/,DT/1.000E-04/,ANGOLD/225./
      REAL ANGL(4)/0.0,30.0,-45.0,88.8/
      REAL FREQ(4)/500.0,  0.0,800.0,300.0/
      DATA NE/1/,ND/2/,NIT/49950/,NP/300/,MP/128/,IT/0/,
     *N1/1/,NT/1/,NA/1/,NF/1/,NA1/1/,NF1/1/
      DATA XMIN/0.0/,XMAX/1000.0/,YMAX/20.0/,FMIN/0.0/,
     *FMAX/1000.0/,YMIN/-90.0/,ZERO/0.0/,TEN/10.0/
C
CC        INITIALIZE GRAPH TITLES WITH NOMINAL VALUES FOR
CC        PARAMETERS.
C
      REAL IT1(24)/'ADAPTIVE FILTER  FO= 500,BW= 200',
     *',O=  0.0,T=   00OUTPUT SPECTRUM ',
     *'      JOHNSON BOX 70      1 AUG 69'/
      REAL IT2(24)/'ADAPTIVE FILTER  FO= 500,BW= 200',
     *',O=  0.0,T=   00FREQUENCY RESPON',
     *'SE   JOHNSON BOX 70      1 AUG 69'/
      REAL IT3(24)/'ADAPTIVE FILTER  FO= 500,BW= 200',
     *',F=  500,T=   00DIRECTIVITY PATT',
     *'ERN  JOHNSON BOX 70      1 AUG 69'/
      REAL IT4(4)/'          45.0    '/,LAB/'    '/
      REAL IT5(4)/' 500              '/
      END
C     MAIN PROGRAM
C-----------------------------------------------------------
C
C           IN ADDITION TO THE FILTER EQUATIONS, THE MAIN
C        PROGRAM CONTAINS THE PROCEDURES FOR GETTING THE 2250
C        DISPLAY UNIT ON LINE, DEFINING THE GRAPHIC DATA SETS
C        TO BE DISPLAYED, ALLOWING INPUT/OUTPUT BETWEEN THE
C        DISPLAY UNIT AND THE PROGRAM, AND HANDLING INTERRUPTS
C        FROM THE 2250 PROGRAMMED FUNCTION KEYS.
C
C-----------------------------------------------------------
      COMMON/VALUES/A,D,X(11,48),W(11,11),DESIG,SIGOUT,SIG,
```

```
     *DT,SPACE,FO,BW,C,DF,KS,ANG,T1,IT1,IT2,IT3,IT4,IT5,LAB,
     *ANGL(4),FREQ(4),XP(128),XP2(505),YP2(505),TEMP(15),
     *S(64),ANGOLD,XY(256,4),JXY(4)
      COMMON/SIZES/NE,ND,NIT,NCPLX,NF,NA,NT,NN,KDEL,INV(64),
     *IT,N1,NNJ,ICODE2,MP,NP,NA1,NF1,NULL(1)
      COMMON/IPLOTS/IPLOT1,IPLOT2,IPLOT3,IPLOT4,IATL,ITEXT1,
     *ITEXT2,IATN1,IKEY1
      COMMON/SCALES/XMIN,XMAX,YMAX,FMIN,FMAX,YMIN,ZERO,TEN
      COMPLEX A(256),D(256)
      REAL IT1(24),IT2(24),IT3(24),IT4(4),IT5(4),LAB,KS
      DIMENSION SIGIN(11,11)
      DEFINE FILE 11(5,100,E,J3)
      NULL(1)=-3
      ICODE=9
      ICODE1=15
      ICODE2=ICODE1
C-------------------------------------------------------------
C         INITIALIZE GRAPHIC SUBROUTINE PACKAGE, GRAPHIC
C         DEVICE, AND GRAPHIC DATA SETS.
C-------------------------------------------------------------
      CALL INGSP(IPLOT,NULL)
      CALL INDEV(IPLOT,10,I2250,512)
      CALL INGDS(I2250,IPLOT1)
      CALL INGDS(I2250,IPLOT2)
      CALL INGDS(I2250,IPLOT3)
      CALL INGDS(I2250,IPLOT4,256)
      CALL INGDS(I2250,ITEXT1,256)
      CALL INGDS(I2250,ITEXT2,256)
C
CC        SET GRAPHIC AND CHARACTER MODES.
C
      CALL SDATM(ITEXT1,3)
      CALL SCHAM(ITEXT1,4)
      CALL SCHAM(ITEXT2,4)
C
CC        SET GRAPHIC DATA SET LIMITS.
C
      CALL SGDSL(IPLOT1,1.0,1.0,11.0,11.0,0.0,0.0,12.0,12.0)
      CALL SGDSL(IPLOT2,1.0,1.0,11.0,11.0,0.0,0.0,12.0,12.0)
      CALL SGDSL(IPLOT3,0.0,0.0,12.0,12.0,0.0,0.0,12.0,12.0)
      CALL SGDSL(IPLOT4,1.0,1.0,11.0,11.0,0.0,0.0,12.0,12.0)
      CALL SGDSL(ITEXT2,1.0,0.0,11.0,10.0,0.0,0.0,12.0,12.0)
C
CC        SET DATA LIMITS.
C
      CALL SDATL(IPLOT1,XMIN,ZERO,XMAX,YMAX)
      CALL SDATL(IPLOT2,FMIN,YMIN,FMAX,TEN)
      CALL SDATL(IPLOT3,-1.2,-1.2,1.2,1.2)
```

```
      CALL SDATL(IPLOT4,0.0,0.0,10.0,10.0)
      CALL SDATL(ITEXT1,0,0,50,35)
      CALL SDATL(ITEXT2,0.0,0.0,10.0,10.0)
C
CC        SET SCISSCRING CPTICNS.
C
      CALL SSCIS(ITEXT1,3)
      CALL SSCIS(ITEXT2,3)
C
CC        ENABLE ATTENTIONS FRCM SPECIFIED PROGRAMMED FUNCTION
CC        KEYS AND FROM THE KEYBOARD 'END' KEY.
C
      CALL CRATL(I2250,IATL)
      CALL ENATN(IATL,1,-3,8,9,15,21,22,27,-32)
C
CC        GENERATE A 10 INCH BY 10 INCH GRID.
C
      DO 86 I=1,11
      X1=I-1
      CALL STPCS(IPLOT4,X1,0.0)
      CALL PLINE(IPLOT4,X1,10.0)
      Y=X1
      CALL STPOS(IPLOT4,0.0,Y)
   86 CALL PLINE(IPLOT4,10.0,Y)
C-----------------------------------------------------------
C         ENTER DESIRED FILTER PARAMETERS FOR THIS RUN.
C-----------------------------------------------------------
CC
CC        DISPLAY NCMINAL GRAPH TITLE.
C
   87 CALL PTEXT(ITEXT1,IT1(1),20,NULL,NULL,1,0,31)
      CALL PTEXT(ITEXT1,IT1(6),28,1)
      CALL PTEXT(ITEXT1,IT1(13),36,11,NULL,1,0,30)
      CALL PTEXT(ITEXT1,IT1(22),12,2)
      CALL EXEC(ITEXT1)
C
CC        INSERT CURSOR INTO DISPLAY SC THAT NEW DATA MAY BE
CC        ENTERED.
C
      CALL ICURS(ITEXT1,1)
C
CC        WAIT UNTIL ATTENTICN FRCM 'END' KEY INDICATES THAT
CC        THE DATA HAS BEEN ENTERED.
C
      CALL RQATN(IATL,IATN5,2,NULL,32)
C
CC        MOVE CURSCR TO NEXT LINE AND WAIT FOR AN ATTENTION
C         FROM THE 'END' KEY.
```

```
C
      CALL ICURS(ITEXT1,2)
      CALL RQATN(IATL,IATN5,2,NULL,32)
C
CC       READ NEW PARAMETERS INTO GRAPH TITLE.
C
      CALL GSPRD(ITEXT1,IT1(6),28,1,NULL,1)
      CALL GSPRD(ITEXT1,IT1(22),12,1,NULL,2)
      CALL GSPRD(ITEXT1,IT1(13),8,1,NULL,11)
C
CC       CHANGE CORRESPONDING PARAMETERS IN THE OTHER TWO
CC       GRAPH TITLES.
C
      DO 88 I=6,8,2
      IT2(I)=IT1(I)
   88 IT3(I)=IT1(I)
      DO 89 I=22,24
      IT2(I)=IT1(I)
   89 IT3(I)=IT1(I)
C
CC       CHANGE NOMINAL INPUT VALUES INTO 'A' FORMAT.
C
      WRITE(11'1,1000) NE,ND,SPACE,C,KS,FO,BW,ANG,NT
      READ(11'1,1001) TEMP
C
CC       DISPLAY REQUEST FOR INPUT DATA USING NOMINAL VALUES
C
      CALL PTEXT(ITEXT1,'# OF ELEMENTS          =',22,NULL,
     *NULL,1,1,26)
      CALL PTEXT(ITEXT1,TEMP(1),4,3)
      CALL PTEXT(ITEXT1,'# OF DELAYS            =',22,NULL,
     *NULL,1,1,24)
      CALL PTEXT(ITEXT1,TEMP(2),4,4)
      CALL PTEXT(ITEXT1,'EACH DELAY =       DT *',22,NULL,
     *NULL,1,1,22)
      CALL PTEXT(ITEXT1,TEMP(15),4,12)
      CALL PTEXT(ITEXT1,'ELEMENT SPACING        =',22,NULL,
     *NULL,1,1,20)
      CALL PTEXT(ITEXT1,TEMP(3),8,5)
      CALL PTEXT(ITEXT1,'SPEED OF SOUND         =',22,NULL,
     *NULL,1,1,18)
      CALL PTEXT(ITEXT1,TEMP(5),8,6)
      CALL PTEXT(ITEXT1,'CONVERGENCE CONSTANT =',22,NULL,
     *NULL,1,1,16)
      CALL PTEXT(ITEXT1,TEMP(7),8,7)
      CALL PTEXT(ITEXT1,'CENTER FREQUENCY       =',22,NULL,
     *NULL,1,1,14)
      CALL PTEXT(ITEXT1,TEMP(9),8,8)
```

```
      CALL PTEXT(ITEXT1,'BANDWIDTH                =',22,NULL,
     *NULL,1,1,12)
      CALL PTEXT(ITEXT1,TEMP(11),8,9)
      CALL PTEXT(ITEXT1,'DESIRED LOOK ANGLE    =',22,NULL,
     *NULL,1,1,10)
      CALL PTEXT(ITEXT1,TEMP(13),8,10)
      CALL EXEC(ITEXT1)
      CALL ICURS(ITEXT1,3)
      CALL RQATN(IATL,IATN5,2,NULL,32)
      CALL ICURS(ITEXT1,4)
      CALL RQATN(IATL,IATN5,2,NULL,32)
      CALL ICURS(ITEXT1,12)
      CALL RQATN(IATL,IATN5,2,NULL,32)
      DO 90 I=5,10
      CALL ICURS(ITEXT1,I)
   90 CALL RQATN(IATL,IATN5,2,NULL,32)
C
CCC      READ INPUT VALUES IN 'A' FORMAT.
C
      CALL GSPRD(ITEXT1,TEMP(1),4,1,NULL,3)
      CALL GSPRD(ITEXT1,TEMP(2),4,1,NULL,4)
      CALL GSPRD(ITEXT1,TEMP(15),4,1,NULL,12)
      IND=3
      DO 91 I=5,10
      CALL GSPRD(ITEXT1,TEMP(IND),8,1,ITRM,I)
   91 IND=IND+2
C
CCC      CHANGE VALUES INTO PROPER FORMAT.
C
      WRITE(11'1,1001) TEMP
      READ(11'1,1000) NE,ND,SPACE,C,KS,FO,BW,ANG,NT
C
CCC      WRITE FILTER PARAMETERS ON PRINTED OUTPUT.
C
      WRITE(6,2000) IT,NE,SPACE
      WRITE(6,2001) ND,NT,DT
      WRITE(6,2002) C
C
CCCCC      DISPLAY REQUEST FOR INITIAL PARAMETERS FOR THE
      'FREQUENCY RESPONSE' AND 'DIRECTIVITY PATTERN'
      PLOTS FOR THIS RUN.
C
      CALL PLTDAT(692)
C
CCCC      GENERATE AND DISPLAY 'ANGLE' AND 'TIME', USING
      NOMINAL VALUES.
C
   92 CALL PTEXT(ITEXT2,'ANGLE =',7,NULL,NULL,1,1.0,0.3)
```

```
      CALL PTEXT(ITEXT2,TEMP(13),8,1)
      CALL PTEXT(ITEXT2,'TIME =',6,NULL,NULL,1,6.0,0.3)
      CALL PTEXT(ITEXT2,'0000',4,2,IKEY1,1,7.5,0.3)
      CALL EXEC(ITEXT2)
C
CC       DISPLAY GRID LINES.
C
      CALL EXEC(IPLOT4)
C-----------------------------------------------------------
C                 INITIALIZE FILTER EQUATIONS
C-----------------------------------------------------------
      IFF=0
      IP=1
      NNJ=ND+1
      NJ=NNJ*NT
      L=NJ
      NJP=NJ+1
      KI=NJP
      NE02=NE/2
C
CC       SYNTHESIZE DESIRED SIGNAL.
C
      CALL SIGREF(67)
C
CC       CALCULATE TIME DELAY BETWEEN ELEMENTS REQUIRED TO
CC       OBTAIN THE DESIRED LOOK ANGLE.
C
    7 CALL DIRECT
C
CC       CALCULATE X AXIS ARRAY FOR 'OUTPUT SPECTRUM' PLOT.
C
      DO 8 J=1,MF
    8 XP(J)=(J-1)*DF
C
CC       INITIALIZE SIMULATED DELAY LINE.
C
      SIGMA=0.707107
      IX=1172903975
      DO 9 J=1,NJ
      LL=NJP-J
C     READ(9) (X(I,LL),I=1,11)
      DO 9 I=1,11
    9 CALL GAUSS(IX,SIGMA   ,0.0,X(I,LL))
C-----------------------------------------------------------
C                 START OF FILTER EQUATIONS
C-----------------------------------------------------------
      DO 5 M=1,NIT
C     READ(9) (X(I,L),I=1,11)
```

```
      DO 785 I=1,11
  785 CALL GAUSS(IX,SIGMA    ,0.0,X(I,L))
      SIG=0.0
      DESIG=0.0
      SIGOUT=0.0
      KREF=KI-NEC2*KDEL
      IF(KREF.LT.1) KREF=KREF+NN
      IF(KREF.GT.NN) KREF=KREF-NN
      DO 3 I=1,NE
      K=KI
      DO 2 J=1,NNJ
      JJ=L+(J-1)*NT
      IF(JJ.GT.NJ) JJ=JJ-NJ
      REALD=D(K)
      SIGIN(I,J)=X(I,JJ)+REALD
      SIG=SIG+W(I,J)*SIGIN(I,J)
      K=K-NT
      IF(K.LT.1) K=K+NN
    2 CONTINUE
      KI=KI-KDEL
      IF(KI.LT.1) KI=KI+NN
      IF(KI.GT.NN) KI=KI-NN
    3 CONTINUE
      DESIG=D(KREF)
      DESIG=DESIG*NE
      ERROR=DESIG-SIG
      DO 4 I=1,NE
      DO 4 J=1,NNJ
      JJ=L+(J-1)*NT
      IF(JJ.GT.NJ) JJ=JJ-NJ
      W(I,J)=W(I,J)+KS*ERROR*SIGIN(I,J)
      SIGOUT=SIGOUT+W(I,J)*X(I,JJ)
    4 CONTINUE
      L=L-1
      IF(L.LT.1) L=NJ
      KI=KI+NE*KDEL+1
      IF(KI.GT.NN) KI=KI-NN
      IF(KI.LT.1) KI=KI+NN
C-----------------------------------------------------------
C                 END OF FILTER EQUATIONS
C-----------------------------------------------------------
C
C        SAMPLE OUTPUT SIGNAL FOR SPECTRAL ANALYSIS.
C
      IF(IP.GT.NN ) GO TO 6
      IFF=IFF+1
      A(IFF)    =SIGOUT
C
```

```
C-------------------------------------------------------------
C              BRANCHING AND CONTROL EQUATIONS
C-------------------------------------------------------------
C
C- L1 -- HAVE 'NP' ADDITIONAL ITERATICNS OF THE FILTER
C        EQUATIONS BEEN PERFORMED ? ---------------------->
C
    6 IF(MOD(M,NP)) 5,20,5
C
C------>       NO.  CONTINUE ITERATING FILTER EQUATIONS.
C
C------>       YES.  UPDATE 'TIME' DISPLAY, THEN GO TO L2. ->
C
   20 IT=M/10
      IFF=0
      IP=0
      WRITE(11'2,1002) IT
      READ(11'2,1003)T1
      CALL RESET(ITEXT2,2,IKEY1)
      CALL PTEXT(ITEXT2,T1,4,2,IKEY1,1,7.5,0.3)
      CALL EXEC(ITEXT2)
C
C- L2 -- HAS THERE BEEN AN ATTENTION FROM PFK-8, PFK-9,
C        PFK-15, PFK-21, OR PFK-27 ? ---------------------->
C
      CALL RQATN(IATL,IATN1,1,NULL,8,9,15,21,27)
C
C------>       NO.  GO TO L3. ---------------------------->
C
      IF(IATN1.LT.5) GC TO 103
C
C------>       YES.  IF ATTENTICN WAS FROM : -------------->
C
      ICODE=IATN1
      IF(ICODE-9) 100,101,102
C
C------>             PFK-8.  REMOVE GRID LINES FROM DISPLAY,
C                            THEN GO TO L3. ---------------->
C
  100 CALL OMIT(IPLOT4)
      GO TO 103
C
C------>             PFK-9.  DISPLAY GRID LINES, THEN GO TO
C                            L3. --------------------------->
C
  101 CALL INCL(IPLOT4)
      GO TO 103
C
```

```
C------>            ANY     GRAPH BEING DISPLAYED WILL BE
C                   OTHER.  CHANGED.  CALCULATE AND DISPLAY
C                           PROPER SCALE FOR NEW GRAPH.
C                           UPDATE 'ICODE1',THEN GO TO L3. ->
C
  102 ICODE1=IATN1
      CALL SSCALE
C
C- L3 -- IF 'ICODE1' INDICATES THAT PREVIOUS ATTENTION WAS
C        FROM : ------------------------------------------>
C
  103 IF(ICODE1-21) 104,105,106
C
C------>        PFK-15.  CALCULATE AND DISPLAY 'OUTPUT SPEC-
C                        TRUM', THEN GO TO L4. ------------->
C
  104 CALL SPECTR(&107)
C
C------>        PFK-21.  CALCULATE AND DISPLAY 'FREQUENCY RES-
C                        PONSE', THEN GO TO L4. ------------->
C
  105 CALL FREQCY(&107)
C
C------>        PFK-27.  CALCULATE AND DISPLAY 'DIRECTIVITY
C                        PATTERN', THEN GO TO L4. ---------->
C
  106 CALL PATERN(&107)
C
C- L4 -- HAS THERE BEEN AN ATTENTION FROM PFK-29, PFK-30,
C        OR PFK-31 ? ------------------------------------->
C
  107 CALL RCATN(IATL,IATN2,1,NULL,29,30,31)
C
C------>        NO.  GO TO L5. ------------------------------>
C
      IF(IATN2.LT.5) GO TO 111
C
C------>        YES.  IF ATTENTION WAS FROM : --------------->
C
      IF(IATN2-30) 108,109,110
C
C------>            PFK-29.  READ NEW DESIRED LOOK ANGLE,
C                            CALCULATE DELAY REQUIRED BETWEEN
C                            ELEMENTS, THEN GO TO L5. ------>
C
  108 CALL DIRECT
      GO TO 111
C
```

```
C------>            PFK-30.  READ NEW PARAMETERS FOR
CCC                          'FREQUENCY RESPONSE' AND
CCC                          'DIRECTIVITY PATTERN' GRAPHS,
CCC                          THEN GO TO L5. ---------------->
C
  109 CALL PLTDAT(&111)
C
C------>            PFK-31.  READ DESIRED SCALE AND SET THE
CCC                          APPROPRIATE SCALING FACTORS,
CCC                          THEN GO TO L5. ---------------->
C
  110 CALL CSCALE
C
CC- L5 -- HAS THERE BEEN AN ATTENTION FROM PFK-1, PFK-22, OR
C         PFK-28 ? ---------------------------------------->
C
  111 CALL RQATN(IATL,IATN3,1,NULL,1,22,28)
C
C------>       NO.  CONTINUE ITERATING FILTER EQUATIONS.
C
      IF(IATN3.EQ.0) GO TO 5
C
C------>       YES.  IF ATTENTION WAS FROM : ------------->
C
      IF(IATN3-22) 113,117,112
C
C------>            PFK-1.   GO TO L6. ------------------->
CCC
C------>            PFK-22.  GO TO L7. ------------------->
CCC
C------>            PFK-28.  READ NEW CENTER FREQUENCY, BAND-
CCC                          WIDTH, AND CONVERGENCE CONSTANT;
CCC                          SYNTHESIZE NEW DESIRED SIGNAL,
CCC                          THEN CONTINUE ITERATING FILTER
CCC                          EQUATIONS.
C
  112 CALL SIGREF(&5)
C
CC- L6.1- HAS LAST 'DIRECTIVITY PATTERN' GRAPH FOR THIS TIME
C         STEP BEEN PLOTTED ? ---------------------------->
CCC
C------>       YES.  GO TO L6. ---------------------------->
CCC
C------>       NO.  CALCULATE,DISPLAY,AND PLOT NEXT GRAPH.
C
  121 NF1=NF1+1
      IF(NF1.LE.NF) GO TO 1115
      GO TO 113
```

```
C
C- L6.2- HAS LAST 'FREQUENCY RESPONSE' GRAPH FOR THIS TIME
C        STEP BEEN PLOTTED ? ------------------------------>
C
C------>       YES.  GO TO L6. ------------------------------>
C
C------>       NO.  CALCULATE,DISPLAY,AND PLOT NEXT GRAPH.
C
  122 NA1=NA1+1
      IF(NA1.LE.NA) GO TO 1116
C
C- L6 -- TURN ON LIGHT IN PFK-1 AND WAIT UNTIL AN ATTENTION
C        OCCURS FROM PFK-1, PFK-2, PFK-3, OR 'END' KEY. --->
C
  113 CALL MLITS(IATL,4,1)
      CALL RQATN(IATL,IATN4,2,NULL,1,2,3,32)
C
C------> IF ATTENTION IS FROM : --------------------------->
C
      IF(IATN4.EQ.32) GO TO 118
      IF(IATN4-2) 114,116,115
C
C------>            PFK-1.   IF GRAPH PRESENTLY DISPLAYED IS
C                             'OUTPUT SPECTRUM', GENERATE DATA
C                             FOR CALCOMP PLOT OF SAME.
C                             OTHERWISE, CALCULATE AND DISPLAY
C                             'OUTPUT SPECTRUM', THEN GENERATE
C                             DATA FOR CALCOMP PLOT OF SAME.
C                             RETURN TO L6. ------------------>
C
  114 IF(ICODE2.EQ.15) GO TO 1124
 1114 CALL SPECTR(&1124)
 1124 CALL PSPECT(&113)
C
C------>            PFK-3.   IF GRAPH PRESENTLY DISPLAYED IS
C                             'DIRECTIVITY PATTERN', GENERATE
C                             DATA FOR CALCOMP PLOT OF SAME.
C                             OTHERWISE, CALCULATE AND DISPLAY
C                             'DIRECTIVITY PATTERN', THEN
C                             GENERATE DATA FOR CALCOMP PLOT
C                             OF SAME.  RETURN TO L6.1. ------>
C
  115 IF(ICODE2.EQ.27) GO TO 1125
 1115 CALL PATERN(&1125)
 1125 CALL PPATRN(&121)
C
C------>            PFK-2.   IF GRAPH PRESENTLY DISPLAYED IS
C                             'FREQUENCY RESPONSE', GENERATE
```

```
C                                DATA FOR CALCOMP PLOT OF SAME.
CC                               CTHERWISE, CALCULATE AND DISPLAY
CCC                              'FREQUENCY RESPONSE', THEN GENER-
CCCC                             ATE DATA FOR CALCOMP PLOT OF
CCCCC                            SAME.  RETURN TO L6.2. --------->
C
  116 IF(ICODE2.EQ.21) GO TO 1126
 1116 CALL FREQCY(&1126)
 1126 CALL PFREQ(&122)
C
C------>              'END'   TURN LIGHT IN PFK-1 OFF, THEN
CC                     KEY.   CONTINUE ITERATING FILTER EQUA-
CCC                           TIONS.
C
  118 CALL MLITS(IATL,2)
      NF1=1
      NA1=1
    5 IP=IP+1
  117 CONTINUE
C 117 REWIND 9
CC
CC- L7 -- HAS THERE BEEN AN ATTENTION FROM THE 'END' KEY ? ->
C
      CALL RQATN(IATL,IATN9,1,NULL,32)
      IF(IATN9.EQ.32) GO TO 200
C
C------>       NO.  REMOVE GRID AND GRAPH FROM DISPLAY, RESET
CC                  PROBLEM AND REQUEST NEW INPUT DATA BY
CCC                 DISPLAYING PRESENT PARAMETERS.
C
      CALL RESET(ITEXT1)
      CALL OMIT(IPLOT4)
      N1=1
      IT=0
      IF(ICODE2-21) 150,151,152
  150 CALL RESET(IPLOT1)
      GO TO 87
  151 CALL RESET(IPLOT2)
      GO TO 87
  152 CALL RESET(IPLOT3)
      GO TO 87
C
C------>       YES.  TERMINATE THE GRAPHIC SUBROUTINE PACKAGE
C
  200 CALL TMGSP(IPLOT)
C-----------------------------------------------------------
CC                     TERMINATE SIMULATION
C-----------------------------------------------------------
```

```
 1000 FORMAT(2I4,F8.3,F8.2,F8.6,3F8.2,I4)
 1001 FORMAT(15A4)
 1002 FORMAT(I4)
 1003 FORMAT(A4)
 2000 FORMAT('1',' T =',I5,'     # OF ELEMENTS =',I3,
     *'     SPACING = ',F8.3)
 2001 FORMAT('C',14X,'# OF DELAYS =',I3,'     DELAY =',I2,
     *' X ',1PE10.4,'  SEC.')
 2002 FORMAT('0',14X,'SPEED OF SOUND = ',F8.3)
      END


      SUBROUTINE SIGREF(*)
C-----------------------------------------------------------------
C
C          SUBROUTINE SIGREF READS NEW VALUES FOR THE
C        PARAMETERS 'CENTER FREQUENCY' AND 'BANDWIDTH', AND
C        SYNTHESIZES THE CORRESPONDING 'DESIRED SIGNAL' BY
C        USE OF THE INVERSE FAST FOURIER TRANSFORM.  IT IS
C        ALSO USED TO READ NEW VALUES OF THE 'CONVERGENCE
C        CONSTANT'.
C
C-----------------------------------------------------------------
      COMMON/VALUES/A,D,X(11,48),W(11,11),DESIG,SIGOUT,SIG,
     *DT,SPACE,FO,BW,C,DF,KS,ANG,T1,IT1,IT2,IT3,IT4,IT5,LAB,
     *ANGL(4),FREQ(4),XP(128),XP2(505),YP2(505),TEMP(15),
     *S(64),ANGOLD,XY(256,4),JXY(4)
      COMMON/SIZES/NE,ND,NIT,NCPLX,NF,NA,NT,NN,KDEL,INV(64),
     *IT,N1,NNJ,ICODE2,MP,NP,NA1,NF1,NULL(1)
      COMMON/IPLOTS/IPLOT1,IPLOT2,IPLOT3,IPLOT4,IATL,ITEXT1,
     *ITEXT2,IATN1,IKEY1
      COMMON/SCALES/XMIN,XMAX,YMAX,FMIN,FMAX,YMIN,ZERO,TEN
      COMPLEX A(256),D(256)
      REAL IT1(24),IT2(24),IT3(24),IT4(4),IT5(4),LAB,KS
      DEFINE FILE 11(5,100,E,J3)
      DIMENSION TEMP1(8)
      IF(N1.EQ.1) GO TO 105
      CALL OMIT(IPLOT4)
      CALL MLITS(IATL,4,28)
      IF(ICODE2-21) 101,102,103
  101 CALL RESET(IPLOT1)
      GO TO 104
  102 CALL RESET(IPLOT2)
      GO TO 104
  103 CALL RESET(IPLOT3)
  104 CALL PTEXT(ITEXT2,'CENTER FREQUENCY =',18,NULL,IKEY,1,
```

```
     *0.3,4.0)
      CALL PTEXT(ITEXT2,TEMP(9),8,3)
      CALL PTEXT(ITEXT2,IT1(6),4,4)
      CALL PTEXT(ITEXT2,'BANDWIDTH =',11,NULL,NULL,1,0.3,3.5)
      CALL PTEXT(ITEXT2,TEMP(11),8,5)
      CALL PTEXT(ITEXT2,IT1(8),4,6)
      CALL PTEXT(ITEXT2,'CONVERGENCE CONSTANT =',22,NULL,NULL,
     *1,0.3,3.0)
      CALL PTEXT(ITEXT2,TEMP(7),8,7)
      CALL EXEC(ITEXT2)
      CALL ICURS(ITEXT2,3)
      CALL RQATN(IATL,IATN6,2,NULL,32)
      CALL ICURS(ITEXT2,5)
      CALL RQATN(IATL,IATN6,2,NULL,32)
      CALL ICURS(ITEXT2,7)
      CALL RQATN(IATL,IATN6,2,NULL,32)
      CALL GSPRD(ITEXT2,TEMP1(1),12,1,ITERM,3,NULL,4)
      CALL GSPRD(ITEXT2,TEMP1(4),12,1,ITERM,5,NULL,6)
      CALL GSPRD(ITEXT2,TEMP1(7),8,1,ITERM,7)
      CALL RESET(ITEXT2,NULL,IKEY)
      CALL MLITS(IATL,2)
      IT1(6)=TEMP1(3)
      IT2(6)=IT1(6)
      IT3(6)=IT1(6)
      IT1(8)=TEMP1(6)
      IT2(8)=IT1(8)
      IT3(8)=IT1(8)
      TEMP(9)=TEMP1(1)
      TEMP(10)=TEMP1(2)
      TEMP(11)=TEMP1(4)
      TEMP(12)=TEMP1(5)
      TEMP(7)=TEMP1(7)
      TEMP(8)=TEMP1(8)
      WRITE(11'2,1000) TEMP1
      READ(11'2,1001) FO,BW,KS
      IF(ICODE2.GT.21) GO TO 105
      CALL INCL(IPLOT4)
  105 CALL CANCEL(2)
      NCPLX=256
      PI=3.141593
      NN=NCPLX
      DF=1.0/(DT*NN)
      NFO=BW/DF+0.5
      RNFO=NFO
      DO 1 M=1,NN
    1 D(M)    =(0.0,0.0)
      FMAG=0.500
      IF(NFO.NE.0) FMAG=SQRT(FMAG/RNFO)
```

```
      NL=F0*DT*NN-NF0/2+1.5
      NU=NL+NFC
      DO 2 K=NL,NU
    2 D(K)     =FMAG
      NL=NN+2-NU
      NU=NL+NFC
      DO 3 K=NL,NU
    3 D(K)     =FMAG
      JXY(1)=1
      JXY(2)=2
      FI=0.
      DO 31 I=1,128
      XY(I,1)=FI
      FI=FI+DF
   31 XY(I,2)=D(I)
      CALL VPLOT(XY,JXY,128,256,1,0,0.0,0.0,0.0,0.0)
      CALL SFRT(D,NCPLX,+1.0)
      WRITE(6,2000) IT,F0,Bw
      WRITE(6,2003) KS
      TT=0.
      DO 33 I=1,256
      XY(I,1)=TT
      TT=TT+DT
      XY(I,2)=D(I)
   33 CONTINUE
      CALL VPLOT(XY,JXY,256,256,1,0,0.0,0.0,0.0,0.0)
   15 FORMAT(5(I4,1P2E10.2))
      RETURN 1
   11 FORMAT( 4(1X,'D(',I4,')= ',2F9.5,4X))
   12 FORMAT(1H1)
   13 FORMAT( I2)
 1000 FORMAT(8A4)
 1001 FORMAT(F8.2,4X,F8.2,4X,F8.6)
 2000 FORMAT('0',' T =',I5,'     CENTER FREQUENCY = ',
     *F8.2,'     BANDWIDTH = ',F8.2)
 2003 FORMAT(' ',14X,'CONVERGENCE CONSTANT = ',F9.6)
      END
```

```
      SUBROUTINE DIRECT
C-----------------------------------------------------------------
C
C             SUBROUTINE DIRECT READS THE 'DESIRED LOCK ANGLE'
C         AND CALCULATES THE REQUIRED TIME DELAY BETWEEN
C         ELEMENTS BASED ON THE VALUES OF THE SPACING BETWEEN
C         ELEMENTS, THE SPEED OF SOUND, AND THE TIME BETWEEN
C         SAMPLES OF THE INPUT SIGNAL. THE DESIRED LOOK ANGLE
C         IS ROUNDED TO THAT ANGLE WITH MAGNITUDE EQUAL TO OR
C         LESS THAN THE INPUT 'DESIRED LOCK ANGLE' WHICH RE-
C         SULTS IN A TIME DELAY BETWEEN ELEMENTS EQUAL TO AN
C         INTEGER MULTIPLE OF THE INPUT SIGNAL SAMPLE PERIOD.
C
C-----------------------------------------------------------------
      COMMON/VALUES/A,D,X(11,48),W(11,11),DESIG,SIGOUT,SIG,
     *DT,SPACE,FO,BW,C,DF,KS,ANG,T1,IT1,IT2,IT3,IT4,IT5,LAB,
     *ANGL(4),FREQ(4),XP(128),XP2(505),YP2(505),TEMP(15),
     *S(64),ANGOLD,XY(256,4),JXY(4)
      COMMON/SIZES/NE,ND,NIT,NCPLX,NF,NA,NT,NN,KDEL,INV(64),
     *IT,N1,NNJ,ICODE2,MP,NP,NA1,NF1,NULL(1)
      COMMON/IPLOTS/IPLOT1,IPLOT2,IPLOT3,IPLOT4,IATL,ITEXT1,
     *ITEXT2,IATN1,IKEY1
      COMMON/SCALES/XMIN,XMAX,YMAX,FMIN,FMAX,YMIN,ZERO,TEN
      COMPLEX A(256),D(256)
      REAL IT1(24),IT2(24),IT3(24),IT4(4),IT5(4),LAB,KS
      DEFINE FILE 11(5,100,E,J3)
      REAL IT6/4H,C= /,IT7/4H,0=-/
      DIMENSION TEMP2(2)
      IF(N1.EQ.1) GO TO 105
      CALL MLITS(IATL,4,29)
      IF(ICODE2-21) 101,102,103                                  REMOVE
  101 CALL RESET(IPLOT1)                                         THESE
      GO TO 104                                                  CARDS
  102 CALL RESET(IPLOT2)                                         WHEN
      GO TO 104                                                  -RCURS-
  103 CALL RESET(IPLOT3)                                         FIXED
  104 CALL ICURS(ITEXT2,1)
      CALL RQATN(IATL,IATN7,2,NULL,32)
      CALL GSPRD(ITEXT2,TEMP2,8,1,NULL,1)
      CALL RCURS(ITEXT2)
      WRITE(11'3,1000) TEMP2
      READ(11'3,1001) ANG
      CALL MLITS(IATL,2)
  105 N1=2
      WMAG=1.0
      IF(ND.NE.0) WMAG=WMAG/ND
      DO 4 I=1,NE
      WINIT=WMAG
```

```
      DO 4 J=1,NNJ
      W(I,J)=WINIT
    4 WINIT=-WINIT
      ARG=C*DT/SPACE
      KDEL=SIN(ANG/57.29578)/ARG+SIGN(0.5,ANG)
      ANG=ARSIN(KDEL*ARG)*57.29578
      WRITE(6,2000) IT,ANG
      WRITE(11'3,1001) ANG
      READ(11'3,1000) TEMP2
      TEMP(13)=TEMP2(1)
      TEMP(14)=TEMP2(2)
      IT1(10)=ABS(ANG)
      WRITE(11'3,1002) IT1(10)
      READ(11'3,1003) IT1(10)
      IF(ANG) 1,2,2
    1 IT1(9)=IT7
      GO TO 3
    2 IT1(9)=IT6
    3 CALL RESET(ITEXT2)
      CALL PTEXT(ITEXT2,'ANGLE =',7,NULL,NULL,1,1.0,0.3)
      CALL PTEXT(ITEXT2,TEMP(13),8,1)
      CALL PTEXT(ITEXT2,'TIME =',6,NULL,NULL,1,6.0,0.3)
      CALL PTEXT(ITEXT2,T1,4,2,IKEY1,1,7.5,0.3)
      CALL EXEC(ITEXT2)
      RETURN
 1000 FORMAT(2A4)
 1001 FORMAT(F8.2)
 1002 FORMAT(F4.1)
 1003 FORMAT(A4)
 2000 FORMAT('0',' T =',I5,'     DESIRED LOOK ANGLE = ',F6.2,
     *' DEGREES')
      END



      SUBROUTINE PLTDAT(*)
C-------------------------------------------------------------
C
C           SUBROUTINE PLTDAT READS THE PARAMETERS FOR WHICH
C        THE CALCOMP PLOTS ARE TO BE MADE.  AT ANY ONE STEP
C        IN TIME, UP TO FOUR 'FREQUENCY RESPONSE' AND FOUR
C        'DIRECTIVITY PATTERN' GRAPHS CAN BE PLOTTED.  THE
C        GRAPH WHICH WILL BE DISPLAYED ON THE 2250 DISPLAY,
C        WILL BE CALCULATED FOR THAT PARAMETER WHICH APPEARS
C        FIRST IN THE PARAMETER LIST FOR THE CORRESPONDING
C        GRAPH. WHILE CALCOMP PLOTS ARE BEING MADE,THE GRAPH
C        BEING PLOTTED WILL BE DISPLAYED SIMULTANEOUSLY ON
```

```
C         THE 2250.
CC
C-----------------------------------------------------------------
      COMMON/VALUES/A,D,X(11,48),W(11,11),DESIG,SIGOUT,SIG,
     *DT,SPACE,FO,BW,C,DF,KS,ANG,T1,IT1,IT2,IT3,IT4,IT5,LAB,
     *ANGL(4),FREQ(4),XP(128),XP2(505),YP2(505),TEMP(15),
     *S(64),ANGOLD,XY(256,4),JXY(4)
      COMMON/SIZES/NE,ND,NIT,NCPLX,NF,NA,NT,NN,KDEL,INV(64),
     *IT,N1,NNJ,ICODE2,MP,NP,NA1,NF1,NULL(1)
      COMMON/IPLOTS/IPLOT1,IPLOT2,IPLOT3,IPLOT4,IATL,ITEXT1,
     *ITEXT2,IATN1,IKEY1
      COMMON/SCALES/XMIN,XMAX,YMAX,FMIN,FMAX,YMIN,ZERO,TEN
      COMPLEX A(256),D(256)
      REAL IT1(24),IT2(24),IT3(24),IT4(4),IT5(4),LAB,KS
      DIMENSION IT7(4)
      DIMENSION ANGLE(8),FREQU(8)
      DEFINE FILE 11(5,100,E,J3)
      IF(N1.EQ.1) GO TO 104
      CALL OMIT(IPLOT4)
      CALL MLITS(IATL,4,30)
      IF(ICODE2-21) 101,102,103
  101 CALL RESET(IPLOT1)
      GO TO 104
  102 CALL RESET(IPLOT2)
      GO TO 104
  103 CALL RESET(IPLOT3)
  104 CALL RESET(ITEXT1)
      CALL PTEXT(ITEXT1,'FREQUENCY RESPONSE PLOT DATA',28,
     *NULL,NULL,1,10,30)
      WRITE(11'4,1000) NA,ANGL
      READ(11'4,1001) NAN,ANGLE
      CALL PTEXT(ITEXT1,'# OF ANGLES =',13,NULL,NULL,1,1,26)
      CALL PTEXT(ITEXT1,NAN,4,1)
      CALL PTEXT(ITEXT1,'ANGLES =',8,NULL,NULL,1,1,24)
      CALL PTEXT(ITEXT1,ANGLE,32,2)
      CALL PTEXT(ITEXT1,'PATTERN PLOT DATA',17,NULL,NULL,1,
     *16,16)
      WRITE(11'5,1000) NF,FREQ
      READ(11'5,1001) NFR,FREQU
      CALL PTEXT(ITEXT1,'# OF FREQUENCIES =',18,NULL,NULL,
     *1,1,12)
      CALL PTEXT(ITEXT1,NFR,4,3)
      CALL PTEXT(ITEXT1,'FREQUENCY =',11,NULL,NULL,1,1,10)
      CALL PTEXT(ITEXT1,FREQU,32,4)
      CALL EXEC(ITEXT1)
      DO 105 N=1,4
      CALL ICURS(ITEXT1,N)
  105 CALL ROATN(IATL,IATN8,2,NULL,32)
```

```
      CALL GSPRD(ITEXT1,NAN,4,1,IRRD,1)
      CALL GSPRD(ITEXT1,ANGLE,32,1,IRRD,2)
      CALL GSPRD(ITEXT1,NFR,4,1,IRRD,3)
      CALL GSPRD(ITEXT1,FREQU,32,1,IRRD,4)
      CALL RESET(ITEXT1)
      CALL MLITS(IATL,2)
      WRITE(11'4,1C01) NAN,ANGLE
      READ(11'4,1000) NA,ANGL
      WRITE(11'5,1C01) NFR,FREQU
      READ(11'5,10C0) NF,FREQ
      DO 106 N=1,4
      IT7(N)=FREQ(N)
  106 IT4(N)=ABS(ANGL(N))
      WRITE(11'4,1C02) IT4
      READ(11'4,1003) IT4
      WRITE(11'5,1C04) IT7
      READ(11'5,10C3) IT5
      IATN1=ICCDE2
      CALL SSCALE
      RETURN 1
 1000 FORMAT(I4,4F8.2)
 1001 FORMAT(A4,8A4)
 1002 FORMAT(4F4.1)
 1003 FORMAT(4A4)
 1004 FORMAT(4I4)
      END


      SUBROUTINE SSCALE
C-------------------------------------------------------------
C
C            SUBRCUTINE SSCALE CALCULATES AND DISPLAYS ON
C         THE 2250 SCALES FOR THE 'FREQUENCY RESPONSE' AND FOR
C         THE 'CUTPUT SPECTRUM' GRAPHS.  ALL SCALES ARE DIS-
C         PLAYED IN INTEGER NUMBERS, HOWEVER THE MAXIMUM
C         ORDINATE FOR THE 'CUTPUT SPECTRUM' PLOT IS MULTI-
C         PLIED INTERNALLY BY A SCALE FACTCR TO ACHIEVE THE
C         DESIRED RANGE.
C
C-------------------------------------------------------------
      COMMON/VALUES/A,D,X(11,48),W(11,11),DESIG,SIGOUT,SIG,
     *DT,SPACE,FO,BW,C,DF,KS,ANG,T1,IT1,IT2,IT3,IT4,IT5,LAB,
     *ANGL(4),FREQ(4),XP(128),XP2(505),YP2(505),TEMP(15),
     *S(64),ANGOLD,XY(256,4),JXY(4)
      COMMON/SIZES/NE,ND,NIT,NCPLX,NF,NA,NT,NN,KDEL,INV(64),
     *IT,N1,NNJ,ICCDE2,MP,NP,NA1,NF1,NULL(1)
```

```
      COMMON/IPLOTS/IPLOT1,IPLOT2,IPLOT3,IPLOT4,IATL,ITEXT1,
     *ITEXT2,IATN1,IKEY1
      COMMON/SCALES/XMIN,XMAX,YMAX,FMIN,FMAX,YMIN,ZERO,TEN
      COMPLEX A(256),D(256)
      REAL IT1(24),IT2(24),IT3(24),IT4(4),IT5(4),LAB,KS
      DIMENSION ITEMP(4)
      DEFINE FILE 11(5,100,E,J3)
      IF(IATN1-21) 1,2,4
    1 ITEMP(1)=YMAX*100.01
      ITEMP(2)=ZERO
      ITEMP(3)=XMIN
      ITEMP(4)=XMAX
      WRITE(11'4,1000) ITEMP
      READ(11'4,1001) ITEMP
      GO TO 3
    2 ITEMP(1)=TEN
      ITEMP(2)=YMIN
      ITEMP(3)=FMIN
      ITEMP(4)=FMAX
      WRITE(11'4,1000) ITEMP
      READ(11'4,1001) ITEMP
    3 CALL RESET(ITEXT1)
      CALL PTEXT(ITEXT1,ITEMP(1),4,1,NULL,1,0,32)
      CALL PTEXT(ITEXT1,ITEMP(2),4,2,NULL,1,0,3)
      CALL PTEXT(ITEXT1,ITEMP(3),4,3,NULL,1,3,2)
      CALL PTEXT(ITEXT1,ITEMP(4),4,4,NULL,1,45,2)
      CALL EXEC(ITEXT1)
      CALL INCL(IPLOT4)
      GO TO 5
    4 CALL RESET(ITEXT1)
      CALL OMIT(IPLOT4)
    5 RETURN
      ENTRY CSCALE
C-------------------------------------------------------------
C
C              ENTRY CSCALE PERMITS THE SCALES OF THE GRAPHS
C         ON THE 2250 TO BE CHANGED TO EXPAND OR COMPRESS THE
C         PLOTS AS DESIRED. THE NEW SCALES ARE READ FROM THE
C         2250 AND THE DATA LIMITS FOR THE GRAPHIC DATA SET
C         ARE CHANGED ACCORDINGLY.  NOTE.  THIS DOES NOT
C         CHANGE THE SCALING OF THE CALCOMP PLOTS.
C
C-------------------------------------------------------------
      IF(ICODE2.GT.21) GO TO 9
      CALL MLITS(IATL,4,31)
      IF(ICODE2-21) 101,102,103                                  REMOVE
  101 CALL RESET(IPLOT1)                                         THESE
      GO TO 104                                                  CARDS
```

```
  102 CALL RESET(IPLOT2)                                                 WHEN
      GO TO 104                                                          -RCURS-
  103 CALL RESET(IPLOT3)                                                 FIXED
  104 CALL ICURS(ITEXT1,1)
      CALL RQATN(IATL,IATN10,2,NULL,32)
      CALL ICURS(ITEXT1,2)
      CALL RQATN(IATL,IATN10,2,NULL,32)
      CALL ICURS(ITEXT1,3)
      CALL RQATN(IATL,IATN10,2,NULL,32)
      CALL ICURS(ITEXT1,4)
      CALL RQATN(IATL,IATN10,2,NULL,32)
      CALL RCURS(ITEXT1)
      CALL GSPRD(ITEXT1,ITEMP(1),8,1,IRRD,1,NULL,2)
      CALL GSPRD(ITEXT1,ITEMP(3),8,1,IRRD,3,NULL,4)
      WRITE(11'5,1001) ITEMP
      IF(ICODE2.GT.15) GO TO 7
    6 READ(11'5,1002) YMAX,ZERO,XMIN,XMAX
      YMAX=YMAX*0.01
      CALL SDATL(IPLOT1,XMIN,ZERO,XMAX,YMAX)
      GO TO 8
    7 READ(11'5,1002) TEN,YMIN,FMIN,FMAX
      CALL SDATL(IPLOT2,FMIN,YMIN,FMAX,TEN)
    8 CALL MLITS(IATL,2)
    9 RETURN
 1000 FORMAT(4I4)
 1001 FORMAT(4A4)
 1002 FORMAT(4F4.0)
      END



      SUBROUTINE SPECTR(*)
C-------------------------------------------------------------
C
C            SUBROUTINE SPECTR CALCULATES AND DISPLAYS THE
C        'OUTPUT SPECTRUM' GRAPH USING THE FAST FOURIER
C        TRANSFORM TO OBTAIN THE FREQUENCY SPECTRUM OF THE
C        FILTER OUTPUT SIGNAL.
C
C-------------------------------------------------------------
      COMMON/VALUES/A,D,X(11,48),W(11,11),DESIG,SIGOUT,SIG,
     *DT,SPACE,FO,BW,C,DF,KS,ANG,T1,IT1,IT2,IT3,IT4,IT5,LAB,
     *ANGL(4),FREQ(4),XP(128),XP2(505),YP2(505),TEMP(15),
     *S(64),ANGOLD,XY(256,4),JXY(4)
      COMMON/SIZES/NE,ND,NIT,NCPLX,NF,NA,NT,NN,KDEL,INV(64),
     *IT,N1,NNJ,ICODE2,MP,NP,NA1,NF1,NULL(1)
      COMMON/IPLOTS/IPLOT1,IPLOT2,IPLOT3,IPLOT4,IATL,ITEXT1,
```

```
     *ITEXT2,IATN1,IKEY1
      COMMON/SCALES/XMIN,XMAX,YMAX,FMIN,FMAX,YMIN,ZERO,TEN
      COMPLEX A(256),D(256)
      REAL IT1(24),IT2(24),IT3(24),IT4(4),IT5(4),LAB,KS
      DEFINE FILE 11(5,100,E,J3)
      IF(ICODE2-21) 103,102,101
  101 CALL RESET(IPLOT3)
      GO TO 103
  102 CALL RESET(IPLOT2)
  103 CALL SFRT(A,NCPLX,-1.0)
      DO 2 I33=1,NCPLX
    2 A(I33)=A(I33)/NCPLX
      DO 7 II=1,MP
    7 YP2(II)=CABS(A(II)     )**2
      CALL RESET(IPLOT1)
      CALL STPCS(IPLOT1,0.0,0.0)
      CALL PLINE(IPLOT1,XP,YP2,NULL,NULL,1,MP)
      CALL EXEC(IPLOT1)
      ICODE2=15
      RETURN 1
      ENTRY PSPECT(*)
C-------------------------------------------------------------
C
C              ENTRY PSPECT GENERATES THE DATA FOR THE 'OUTPUT
C         SPECTRUM' CALCOMP PLOT.  IT ALSO WRITES THE VALUES
C         OF THE ADAPTIVE WEIGHTS ON THE PRINTED OUTPUT.
C
C-------------------------------------------------------------
      WRITE(6,12) IT
      WRITE(6,11) ((I,J,W(I,J),J=1,NNJ),I=1,NE)
      CALL MLITS(IATL,2)
      IT1(12)=T1
      DO 1 I=1,MP
      XY(I,1)=XP(I)
    1 XY(I,2)=YP2(I)
      CALL VPLOT(XY,JXY,MP,256,1,0,0.0,0.0,0.0,0.0,0.0)
      CALL DRAW(MP,XP,YP2,0,0,LAB,IT1,0,0,0,0,0,0,7,5 ,1,LAST)
      RETURN 1
   11 FORMAT( 5(1X,'W(',I2,',',I2,')= ',F10.5,4X))
   12 FORMAT(//,'  T =',I5,//)
      END
```

```
      SUBROUTINE FREQCY(*)
C-----------------------------------------------------------------
C
C              SUBROUTINE FREQCY CALCULATES AND DISPLAYS THE
C          'FREQUENCY RESPONSE' GRAPH.  THE FREQUENCY RESPONSE
C          OF THE FILTER IS CALCULATED, USING THE VALUES OF THE
C          WEIGHTS AT THE TIME THAT THE SUBROUTINE IS CALLED,
C          FOR THE ANGLE PARAMETERS LAST READ BY SUBROUTINE
C          PLTDAT.
C
C-----------------------------------------------------------------
      COMMON/VALUES/A,D,X(11,48),W(11,11),DESIG,SIGOUT,SIG,
     *DT,SPACE,FO,BW,C,DF,KS,ANG,T1,IT1,IT2,IT3,IT4,IT5,LAB,
     *ANGL(4),FREQ(4),XP(128),XP2(505),YP2(505),TEMP(15),
     *S(64),ANGOLD,XY(256,4),JXY(4)
      COMMON/SIZES/NE,ND,NIT,NCPLX,NF,NA,NT,NN,KDEL,INV(64),
     *IT,N1,NNJ,ICODE2,MP,NP,NA1,NF1,NULL(1)
      COMMON/IPLOTS/IPLOT1,IPLOT2,IPLOT3,IPLOT4,IATL,ITEXT1,
     *ITEXT2,IATN1,IKEY1
      COMMON/SCALES/XMIN,XMAX,YMAX,FMIN,FMAX,YMIN,ZERO,TEN
      COMPLEX A(256),D(256)
      REAL IT1(24),IT2(24),IT3(24),IT4(4),IT5(4),LAB,KS
      DEFINE FILE 11(5,100,E,J3)
      REAL IT6/4H,C= /,IT7/4H,0=-/
      DIMENSION GRF(505,2)
      EQUIVALENCE(GRF(1,1),XP2(1)),(GRF(1,2),YP2(1))
      IF(ICODE2-21) 101,103,102
  101 CALL RESET(IPLOT1)
      GO TO 103
  102 CALL RESET(IPLOT3)
  103 TWOPI=6.2831853
      DT1=NT*DT
      NEO2=NE/2+1
      LOWF=FMIN+2.
      MAXF=FMAX
      IDF=(MAXF+499-LOWF)/500
      ANG1=ANGL(NA1)/57.29578
      DELT=SPACE*SIN(ANG1)/C
      II=0
      AMPMAX=NE
      DO 2 K=LOWF,MAXF,IDF
      OMEGA=TWOPI*K
      PHI=OMEGA*DT1
      PSI=OMEGA*DELT
      AMPL=C.C
      II=II+1
      DO 1 I=1,NE
      EPSI=(I-NEC2)*PSI
```

```
      DO 1 J=1,NNJ
      FJM1=J-1
      ARGU=-FPSI-FJM1*PHI
    1 AMPL=AMPL+W(I,J)*COS(ARGU)
      AMPLMG=ABS(AMPL)
      XP2(II)=K
      YP2(II)=AMPLMG
    2 CONTINUE
      DO 3 K=1,II
    3 YP2(K)=20.0*ALOG10(YP2(K)/AMPMAX)
      CALL RESET(IPLOT2)
      CALL STPOS(IPLOT2,0.0,0.0)
      CALL PLINE(IPLOT2,XP2,YP2,NULL,NULL,1,II)
      CALL EXEC(IPLOT2)
      ICODE2=21
      RETURN 1
      ENTRY PFREQ(*)
C-------------------------------------------------------------
C
C              ENTRY PFREQ GENERATES THE DATA FOR THE
C         'FREQUENCY RESPONSE' CALCOMP PLOT.
C
C-------------------------------------------------------------
      CALL MLITS(IATL,2)
      IF(ANG1) 4,5,5
    4 IT2(9)=IT7
      GO TO 6
    5 IT2(9)=IT6
    6 IT2(10)=IT4(NA1)
      IT2(12)=T1
      CALL VPLOT(GRF,JXY,II,505,1,0,0.0,0.0,0.0,0.0,0.0)
      CALL DRAW(II,XP2,YP2,0,0,LAB,IT2,0,20.0,4,0,2,0,7,5 ,
     *1,LAST)
      RETURN 1
      END
```

```
      SUBROUTINE PATERN(*)
C-----------------------------------------------------------------
CCC
CCC            SUBRCUTINE PATERN CALCULATES AND DISPLAYS THE
CCC         'DIRECTIVITY PATTERN' GRAPH.  THE DIRECTIVITY PATTERN
CCC         OF THE FILTER IS CALCULATED, USING THE VALUES OF THE
CCC         WEIGHTS AT THE TIME THAT THE SUBROUTINE IS CALLED,
CCC         FOR THE FREQUENCY PARAMETERS LAST READ BY SUB-
CCC         ROUTINE PLTDAT.  PATERN ALSO CALCULATES AND DISPLAYS
CCC         AN ARROW WHICH POINTS IN THE DIRECTION OF THE DESIRED
CCC         LOCK ANGLE OF THE FILTER.  THE BASE OF THE ARROWHEAD
CCC         INDICATES THE DESIRED MAGNITUDE OF THE DIRECTIVITY
CCC         PATTERN IN THE DESIRED LOOK DIRECTION.
CCC
C-----------------------------------------------------------------
      COMMON/VALUES/A,D,X(11,48),W(11,11),DESIG,SIGOUT,SIG,
     *DT,SPACE,FO,BW,C,DF,KS,ANG,T1,IT1,IT2,IT3,IT4,IT5,LAB,
     *ANGL(4),FREQ(4),XP(128),XP2(505),YP2(505),TEMP(15),
     *S(64),ANGOLD,XY(256,4),JXY(4)
      COMMON/SIZES/NE,ND,NIT,NCPLX,NF,NA,NT,NN,KDEL,INV(64),
     *IT,N1,NNJ,ICODE2,MP,NP,NA1,NF1,NULL(1)
      COMMON/IPLCTS/IPLOT1,IPLOT2,IPLOT3,IPLOT4,IATL,ITEXT1,
     *ITEXT2,IATN1,IKEY1
      COMMON/SCALES/XMIN,XMAX,YMAX,FMIN,FMAX,YMIN,ZERO,TEN
      COMPLEX A(256),D(256)
      REAL IT1(24),IT2(24),IT3(24),IT4(4),IT5(4),LAB,KS
      DIMENSION XXP(64),YYP(64)
      DEFINE FILE 11(5,100,E,J3)
      DIMENSION TEMP3(2)
      IF(ICODE2-21) 101,102,103
  101 CALL RESET(IPLOT1)
      GO TO 103
  102 CALL RESET(IPLOT2)
  103 DT1=NT*DT
      NEO2=NE/2+1
      IF(ANG.EQ.ANGOLD) GO TO 6
      PI=3.1415927
      ANGOLD=ANG
      LL=53
      NPT=181
      DBETA=1.745329E-02
      LP=LL+7
      ANG1=ANG/57.29578
      NPTO2=NPT/2+1
      XS=SIN(ANG1)*0.02
      YS=COS(ANG1)*0.02
      DO 4 M=1,LL
      XXP(M)=M*XS
```

```
    4 YYP(M)=M*YS
      XSM=SIN(ANG1-0.26)*0.02
      YSM=COS(ANG1-0.26)*0.02
      XSP=SIN(ANG1+0.26)*0.02
      YSP=COS(ANG1+0.26)*0.02
      DO 5 L=1,3
      LLP=LL+L
      LPM=LP-L
      XXP(LLP)=XXP(LL)-L*XSM
      YYP(LLP)=YYP(LL)-L*YSM
      XXP(LPM)=XXP(LL)-L*XSP
    5 YYP(LPM)=YYP(LL)-L*YSP
      XXP(LP)=XXP(LL)
      YYP(LP)=YYP(LL)
    6 OMEGA=2.0*PI*FREQ(NF1)
      PHI=OMEGA*DT1
      AMPMAX=NE
      DO 2 K=1,NPT
      AMPL=0.0
      BETA=(K-NPTO2)*DBETA
      PSI=OMEGA*SPACE*SIN(BETA)/C
      DO 1 I=1,NE
      FPSI=(I-NEC2)*PSI
      DO 1 J=1,NNJ
      FJM1=J-1
      ARGU=-FPSI-FJM1*PHI
    1 AMPL=AMPL+W(I,J)*CCS(ARGU)
      AMPLMG=ABS(AMPL)
      XP2(K)=AMPLMG*SIN(BETA)
      YP2(K)=AMPLMG*COS(BETA)
    2 CONTINUE
      DO 3 KK=1,NPT
      XP2(KK)=XP2(KK)/AMPMAX
      YP2(KK)=YP2(KK)/AMPMAX
    3 CONTINUE
      CALL RESET(IPLOT3)
      CALL STPOS(IPLOT3,XP2(1),0.0)
      CALL PLINE(IPLOT3,XP2,YP2,NULL,NULL,1,NPT)
      CALL STPOS(IPLOT3,0.0,0.0)
      CALL PLINE(IPLOT3,XXP,YYP,NULL,NULL,1,LP)
      CALL EXEC(IPLOT3)
      ICODE2=27
      RETURN 1
      ENTRY PPATRN(*)
C-----------------------------------------------------------
CCC
CCC          ENTRY PPATRN GENERATES THE DATA FOR THE
C         'DIRECTIVITY PATTERN' CALCOMP PLOT.
```

```
C
C-------------------------------------------------------------
      CALL MLITS(IATL,2)
      IT3(10)=IT5(NF1)
      IT3(12)=T1
      JXY(3)=3
      JXY(4)=4
      DO 8 I=1,NPT
      XY(I,1)=XP2(I)
      XY(I,2)=YP2(I)
      IF(I.GT.LP)GO TO 7
      XY(I,3)=XXP(I)
      XY(I,4)=YYP(I)
      GO TO 8
    7 XY(I,3)=XXP(LP)
      XY(I,4)=YYP(LP)
    8 CONTINUE
      CALL VPLOT(XY,JXY,NPT,256,2,1,-1.1,1.1,0.0,1.616)
      WRITE(6,20)NE,ND,FO,BW,DT
   20 FORMAT(2I10,3E10.3)
      CALL DRAW(NPT,XP2,YP2,1,0,LAB,IT3,0.4,0.4,0,3,2,2,7,5,
     *0,LAST)
      CALL DRAW(LP,XXP,YYP,3,0,LAB,IT3,0.4,0.4,0,3,2,2,7,5,
     *0,LAST)
      RETURN 1
      END



      SUBROUTINE SFRT(A,NCPLX,ISIGN)                                    000
C FAST FOURIER TRANSFORM SUBROUTINE FOR 2**N POINTS,(N INTEGER) BY N. BRENNER
C PARAMETERS:
C A(I)=COMPLEX ARRAY OF NCPLX POINTS WITH:REAL A(I) IN CELL A(2*I-1)
C                                         IMAG A(I) IN CELL A(2*I)
C NCPLX=NUMBER OF COMPLEX POINTS=2**N WHERE N IS A POSITIVE INTEGER
C ISIGN=NUMBER GIVING DIRECTION OF TRANSFORM:IF ISIGN=-1 PERFORM FORWARD
C       TRANSFORM, IF ISIGN=+1 PERFORM INVERSE TRANSFORM.
C       FORWARD TRANSFORM FOLLOWED BY INVERSE TRANSFORM RESULTS IN THE ORIGINAL
C       FUNCTION MULTIPLIED BY NCPLX.
      DIMENSION A(1)                                                    001
      N=2*NCPLX                                                         002
      J=1                                                               003
      DO 11 I=1,N,2                                                     004
      JP1=J+1                                                           005
      IP1=I+1                                                           006
      IF(I-J)2,4,4                                                      007
    2 TR=A(J)                                                           008
```

```
      TI=A(JP1)                                                         009
      A(J)=A(I)                                                         010
      A(JP1)=A(IP1)                                                     011
      A(I)=TR                                                           012
      A(IP1)=TI                                                         013
    4 M=N/2                                                             014
    5 IF(M-J)6,11,11                                                    015
    6 J=J-M                                                             016
      M=M/2                                                             017
      IF(M-2)11,5,5                                                     018
   11 J=J+M                                                             019
      MMAX=2                                                            020
   13 IF(MMAX-N)14,99,99                                                021
   14 INCR=2*MMAX                                                       022
      PIMI=3.14159265/FLOAT(MMAX)                                       023
      DO 21 M=1,MMAX,2                                                  024
      ANG=PIMI*FLOAT(M-1)                                               025
      WR=COS(ANG)                                                       026
      WI=SIN(ANG)                                                       027
      IF(ISIGN)18,99,19                                                 028
   18 WI=-WI                                                            029
   19 DO 21 I=M,N,INCR                                                  030
      J=I+MMAX                                                          031
      JP1=J+1                                                           032
      IP1=I+1                                                           033
      TR=WR*A(J)-WI*A(JP1)                                              034
      TI=WR*A(JP1)+WI*A(J)                                              035
      A(J)=A(I)-TR                                                      036
      A(JP1)=A(IP1)-TI                                                  037
      A(I)=A(I)+TR                                                      038
   21 A(IP1)=A(IP1)+TI                                                  039
      MMAX=INCR                                                         040
      GO TO 13                                                          041
   99 RETURN                                                            042
      END                                                               043
```

```
      SUBROUTINE VPLOT(XY,JXY,N,NDIM,NCUR,ISCALE,XL,XU,YL,YU)                000
C VPLOT IS A PRINTER PLOT RCUTINE
      DIMENSION IGRID(101),XS(11),YS(17),ICHAR(7),XY(1),JXY(1)              001
      DATA ICHAR/1H+,1H*,1H-,1H$,1H=,1H.,1H /                               002
      XMIN=XL                                                               003
      XMAX=XU                                                               004
      YMIN=YL                                                               005
      YMAX=YU                                                               006
      IF(ISCALE.NE.0)GO TO 32                                               007
      XMAX=-1.0E 20                                                         008
      XMIN=-XMAX                                                            009
      YMAX=XMAX                                                             010
      YMIN=XMIN                                                             011
      J2=0                                                                  012
      DO 31 J=1,NCUR                                                        013
      J2=J2+2                                                               014
      JIX=(JXY(J2-1)-1)*NDIM                                                015
      JIY=(JXY(J2)-1)*NDIM                                                  016
      DO 31 I=1,N                                                           017
      IJX=JIX+I                                                             018
      IJY=JIY+I                                                             019
      IF(XY(IJX).GT.XMAX)XMAX=XY(IJX)                                       020
      IF(XY(IJX).LT.XMIN)XMIN=XY(IJX)                                       021
      IF(XY(IJY).GT.YMAX)YMAX=XY(IJY)                                       022
      IF(XY(IJY).LT.YMIN)YMIN=XY(IJY)                                       023
   31 CONTINUE                                                              024
   32 XR=XMAX-XMIN                                                          025
      IF(XR.EQ.0.0)XR=1.0E-20                                               026
      YR=YMAX-YMIN                                                          027
      IF(YR.EQ.0.0)YR=1.0E-20                                               028
      XT=XMAX*XMIN                                                          029
      YT=YMIN*YMAX                                                          030
      IF(XT.LT.0.0)IYAX=100.0*(-XMIN)/XR+1.5                                031
      IF(YT.LE.0.0)IXAX=64.0*YMAX/YR+1.5                                    032
      XINCR=XR/10.0                                                         033
      YINCR=YR/16.0                                                         034
      XS(1)=XMIN                                                            035
      YS(1)=YMAX                                                            036
      DO 46 I=2,11                                                          037
   46 XS(I)=XS(I-1)+XINCR                                                   038
      DO 47 I=2,17                                                          039
   47 YS(I)=YS(I-1)-YINCR                                                   040
      WRITE(6,10)(XS(I),I=1,11)                                             041
      II=1                                                                  042
      KK=0                                                                  043
      DO 146 LINE=1,65                                                      044
      DO 101 J=1,101                                                        045
  101 IGRID(J)=ICHAR(7)                                                     046
```

```
      IF(YT.GT.0.0)GO TO 109                                            047
      IF(LINE.NE.IXAX)GO TO 109                                         048
      DO 105 J=1,101                                                    049
  105 IGRID(J)=ICHAR(6)                                                 050
  109 IF(XT.LT.0.0)IGRID(IYAX)=ICHAR(6)                                 051
      J2=0                                                              052
      DO 125 J=1,NCUR                                                   053
      JC=MOD(J,5)                                                       054
      J2=J2+2                                                           055
      JIX=(JXY(J2-1)-1)*NDIM                                            056
      JIY=(JXY(J2)-1)*NDIM                                              057
      DO 125 I=1,N                                                      058
      IJX=JIX+I                                                         059
      IJY=JIY+I                                                         060
      IPTY=64.0*(YMAX-XY(IJY))/YR+1.5                                   061
      IF(IPTY.GT.65)IPTY=65
      IF(IPTY.LT.1)IPTY=1                                               063
      IF(IPTY.NE.LINE)GO TO 125                                         064
      IPTX=100.0*(XY(IJX)-XMIN)/XR+1.5                                  065
      IF(IPTX.LT.1)IPTX=1                                               066
      IF(IPTX.GT.101)IPTX=101
      IF(JC.NE.0)GO TO 119                                              068
      IGRID(IPTX)=ICHAR(5)                                              069
      GO TO 125                                                         070
  119 IGRID(IPTX)=ICHAR(JC)                                             071
  125 CONTINUE                                                          072
      IF(KK.GT.0)GO TO 134                                              073
      WRITE(6,20)YS(II),(IGRID(I),I=1,101),YS(II)                       074
      II=II+1                                                           075
      GO TO 135                                                         076
  134 WRITE(6,30)(IGRID(I),I=1,101)                                     077
  135 KK=KK+1                                                           078
      IF(KK.NE.4)GO TO 146                                              079
      KK=0                                                              080
  146 CONTINUE                                                          081
      WRITE(6,40)(XS(I),I=1,11)                                         082
      RETURN                                                            083
   10 FORMAT(1H1,1PE15.2,10E10.2/10X,1H*,20(5H+****),2H+*)              084
   20 FORMAT(1PE10.2,1H+,101A1,1H+,E9.2)                                085
   30 FORMAT(10X,1H*,101A1,1H*)                                         086
   40 FORMAT(10X,1H*,20(5H+****),2H+*/1PE16.2,10E10.2)                  087
      END                                                               088
```

BIBLIOGRAPHY

1. Tolstoy, I. and Clay, C.S., Ocean Acoustics, McGraw-Hill Book Co., New York, 1966, p.15.

2. Lamb, H. J., Hydrodynamics, Cambridge University Press, 6th ed., 1932, pp. 258, 259.

3. Collin, R. E., Field Theory of Guided Waves, McGraw-Hill Book Co., New Yrok, N.Y., 1960, p.22.

4. Richtmyer, R. D. and Morton, K. W., Difference Methods for Initial Value Problems, Interscience Publishers, New York, N. Y., 1967, pp. 24-25.

5. Brilloun, L., Wave Propagation in Periodic Structures, McGraw-Hill Book Co., New Yrok, 1946.

6. Lighthill, M. J., Fourier Analysis and Generalized Functions, Cambridge University Press, 1959.

7. Crank, J. and Nicholson, P., A Practical Method For Numerical Integration of Solutions of Partial Differential Equations of Heat Conduction Type, Proc. Cambridge Philos. Soc., vol. 43, p. 50.

8. Hadamard, J., Lectures on Cauchy's Problem in Linear Partial Differential Equations, Yale University Press, 1923.

9. Salvadori, M and Baron, M., Numerical Methods in Engineering, Prentice-Hall, Inc., Englewood Cliffs, N. J., chapt. III.

10. Courant, R., Fredrichs, K. O. and Lewy, H., Uber Die Partiellen Differenzengleichungen der Mathematischen Physik, Mathematichen Annalen, vol. 100, p. 32, 1928.

11. Kreiss, H. O., Uber die Stabilitatsdefinition fur Differenzengleichungen die Partielle Differentialgleichungen Approximieren, Nordisk Tidskr. Informations-Behandling, vol. 2, 1962, p. 153.

12. Kreiss, H. O., Uber implizite Differenzenmethoden fur partielle Differentialgleichchungen, Numer. Math., vol. 5, p. 24.

13. Lax, P. D. and Wendroff, B., On the Stability of Difference Schemes, Comm. Pure Appl. Math., vol. 15, 1964, p. 363.

14. Johannson, O., and Kreiss, H. O., Uber das Verfahren der zentralen Differenzen zur Losung des Cauchy-Problems fur partielle Differentialgleichungen, Nordisk Tidskr, Informations-Behandling, vol. 3, 1963, p. 97.

15. Kato, T., Estimation of Iterated Matrices With Application to the Von Neumann Condition, Numer. Math., vol. 2, 1960, p. 22.

16. Bagrinovskii, K. A. and Godunov, S. K., Difference Schemes for Multi-dimensional Problems, Dokl. Akad. Mauk. USSR, vol. 115, 1957, p. 431.

17. Yanenko, N. N., On Weak Approximation of Systems of Differential Equations, Sibirsk. Mat. Zh., vol. 5, 1964, p. 1430.

18. Douglas, J. and Gunn, J., A General Formulating of Alternating Direction Methods, I., Numer. Math., vol. 6, 1964, p. 428.

19. Cooley, J. W. and Tukey, J. W., An Algorithm for the Machine Calculation of Complex Fourier Series, Math. of Comp., vol. 19, no. 90, 1965, pp. 297-301.

20. Frazer, R., Duncan, W., and Collar, A., Introduction to Elementary Matrices, Cambridge University Press, 1955, p. 138.

21. Courant, R. and Hilbert, D., Methods of Mathematical Physics, Vol. I, New York, Interscience Publishers, Inc., 1962.

22. Szego, G., Orthogonal Polynomials, American Mathematical Society, New York, 1959.

23. Kelvin, Lord, Popular Lectures, vol. I, 1881, p. 185.

24. Watkins, D. A., Theory of Microwave Devices, John Wiley and Sons, Inc., 1958, p. 2.

25. Schelkunoff, S. A., Communications on Pure and Applied Mathematics, D. van Nostrand, 1943, p. 208., vol. 4, p. 117, 1951.

26. Gentleman, W. M. and Sande, G., "Fast Fourier Transforms - For Fun and Profit," IEEE Transactions on Audio and Electroacoustics, June 1967.

27. Stockham, T. G., "High Speed Convolution and Correlation," AFIPS, vol. 28, 1966 Spring Jont Computer Conference, Spartan Books, Washington, 1966.

28. Johnson, R. M., "Graphical Fourier Operations," Research Report no. 31, U. S. Naval Postgraduate School, October 1962.

29. Brenner, N., "Three Fast Fourier Transform Fortran Programs," Lincoln Lab. Technical Report, no. 67-167, M. I. T.

30. Middleton, D. and Groginski, H., "Detection of Random Acoustic Signals by Receiver with Distributed Elements," J. Acoust. Soc. Am., vol. 38, no. 5, November 1965.

31. Capon, J., Greenfield, R. J. and Kolker, R. J., "Multidimensional Maximum-Likelihood Processing of a Large Aperature Seismic Array," Proc. IEEE, 55 (2), 1967.

32. Cox, LCDR, H., USN, "Interrelated Problems in Estimation and Detection I," Proc. Nato Advanced Study Institute on Signal Processing with Emphasis on Underwater Acoustics, August 1968, Enschede, The Netherlands.

33. Burg, J. P., "Three-Dimensional Filtering with an Array of Seismometers," Geophysics, vol. 29, October 1964, pp. 693-713.

34. Claerbout, J. F., "Detection of P Waves from Weak Sources at Great Distances," Geophysics, vol. 29, April 1964, pp. 197-211.

35. Bekey, G. A. and Karplus, W. J., Hybrid Computation, John Wiley and Sons, New York, 1968.

36. Von Neumann, J., Mathematische Grundlogen der Quantumechanik, Dover, 1943.

37. Hildebrand, F. B., "On the Convergence of Numerical Solutions of the Heat Flow Equation," J. Math. Physics, vol. 31, p. 35.

38. Ramo, S., Whinnery, J. R., and Van Duzer, T., Field and Waves in Communication Electronics, John Wiley and Sons, New York, 1965, pp. 29-30

39. Somerfeld, A., Partial Differential Equations, Academic Press, New York, 1949, p. 41.

40. Widrow, B., "Adaptive Antenna Systems," Proc. of the IEEE, vol. 55, no. 12, December 1967.

41. Mantey, P. E., "Convergent Automatic-Synthesis Procedures for Sampled-Data Networks with Feedback," Tech. Report no. 6773-1, Stanford Electronics Laboratories, October 1964.

42. Whittle, P., Prediction and Regulation by Linear Least-Square Methods, Van Nostrand, Princeton, N. J., 1963.

43. Claerbout, J. F., "Digital Filters and Applications to Seismic Detection and Discrimination," Rept. no. 5, Contract AF19(604)7378 M. I. T., Cambridge, Mass., February 1963.

44. Ragazzini, J. R. and Franklin, G. F., Sampled-Data Control Systems, McGraw-Hill Book Co., New York, 1947.

45. Kron, G., Tensor Analysis of Networks, John Wiley and Sons, New York, 1942.

46. Kalman, R.E., "A New Approach to Linear Filtering and Prediction Problems," Trans, ASME, J. Basic Engineering, March 1960.

47. Lee, R. C. K., Optimal Estimation, Identification and Control, Research Monography No. 28, M. I. T. Press, Cambridge, Mass., 1964.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia Attn: IRS	20
2.	Assoc. Prof. H. A. Titus, Code 52Ts Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	7
3.	Dr. Roy M. Johnson Department of Electrical Engineering Montana State University Bozeman, Montana 59715	7
4.	Professor W. P. Cunningham, Code 61Cm Department of Physics Naval Postgraduate School Monterey, California 93940	1
5.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
6.	Naval Ship Systems Command Navy Department Washington, D.C. 20360 Attn: Dave Pope, Code 2126 MUN	2
7.	Dr. Jerry Kenneson Naval Electronics Laboratory Center San Diego, California 92152	2
8.	Mr. Dick Reins Fleet Numerical Weather Central Naval Postgraduate School Monterey, California 93940	1
9.	Professor C. E. Menneken, Code 023 Dean of Research Administration Naval Postgraduate School Monterey, California 93940	2

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Naval Postgraduate School Monterey, California 93940	Unclassified
	2b. GROUP

3. REPORT TITLE

Large-Domain Discrete Integration Techniques for the Wave Equation as an Aid In the Calculation of Propagation Loss and the Study of Adaptive Acoustic Arrays

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Doctors Thesis, October 1969

5. AUTHOR(S) (First name, middle initial, last name)

Roy Martin Johnson

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
October 1969	313	47

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

This work is concerned with two inter-related problems: (1) the theory and application of algorithms for the discrete integration of the acoustic wave equation in large, multi-dimensional, variable-parameter domains and, (2) the automatic synthesis of adaptive acoustic arrays for the detection and estimation of received acoustic signals which are incident from such domains.

It is shown that the steady-state integration of large domains may be partitioned into subdomain integrations for slowly-varying variable-parameter domains. A new method of integration based upon the Fast Fourier Transform is given. A new method for obtaining closed form coefficient expressions for the Fast Fourier Transform is shown and illustrated.

The results of a general computer-display simulation program for the Widrow feed-forward algorithm are given. Several limitations and possible modifications to the Widrow procedure are given. The use of Kalman-type filters as an alternative method is introduced.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Security Classification

A-31408

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Partial-differential equations						
Integration, filtering						
Acoustic arrays